SECURITY AND PRIVACY IN BIOMETRICS-BASED SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Erkam Uzun

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science
College of Computing

Georgia Institute of Technology

August 2021

SECURITY AND PRIVACY IN BIOMETRICS-BASED SYSTEMS

Thesis committee:

Dr. Wenke Lee, Advisor
School of Computer Science
Georgia Institute of Technology

Dr. Mustaque Ahamad
School of Computer Science
Georgia Institute of Technology

Dr. Alexandra Boldyreva
School of Computer Science
Georgia Institute of Technology

Dr. Irfan Essa
School of Interactive Computing
Georgia Institute of Technology

Dr. Vladimir Kolesnikov
School of Computer Science
Georgia Institute of Technology

Dr. Nasir Memon
Tandon School of Engineering
New York University

Date approved: July 7, 2021

Ex oriente lux.

To my beloved wife, Gozde,

and our dear daughter,

Merih Sara.

ACKNOWLEDGMENTS

First and foremost, I want to express my deepest gratitude to my advisor, Dr. Wenke Lee, for his motivation and guidance throughout my studies. In my PhD journey, Wenke always encouraged me to pursue the good research direction and to explore the most challenging problems. I consider him as a role model thanks to his integrity and generosity in addition to his academic advising. I hope to pass all the lessons I learned from him to my future students and advisees.

I would also like to express my thanks to my doctoral dissertation committee members: Dr. Ahamad Mustaque, Dr. Irfan Essa, Dr. Alexandra Boldyreva, Dr. Vladimir Kolesnikov and Dr. Nasir Memon. I also consider myself very fortunate to have a chance to work and collaborate with them. Their insightful comments, suggestions and guidance on my research have not only greatly contributed my thesis but also helped broaden my horizons for my future research. I would like to take the opportunity to thank my other collaborators: Dr. Simon Chung and Carter Yagemann. I have been very lucky to work with Simon as I learned a lot from him throughout my PhD.

I would like to express special thanks to Dr. Husrev Taha Sencar and Dr. Bulent Tavli who led me to where I am today. They started advising me to do high quality research when I was an undergraduate, and motivated me to pursue PhD after graduation.

This thesis could bot be done without the help of many brilliant colleagues. I would like to thank every member of GTISC laboratory and the IISP staffs: Elizabeth Ndongi, Trinh Doan, Sue Jean Chae and Gloria Griessman for their efforts to provide such a friendly research environment.

I also would like to specially thank Abdurrahman Yasar for his fellowship in this journey, and I would like to take the opportunity to thank Semih Sahin, Emre Yilmaz and every other members of Turkish Student Association for their friendship.

Especially, and the most importantly, I would like to thank to my parents Cafer and

Munevver, my sister Zeynep and Zeliha, my brother Emir, my wife Gozde and her parents for their support and unconditional love. I am tremendously grateful for all the selflessness and the sacrifices they have made on my behalf.

Last but not least, I want to devote this thesis to my lovely wife and our dear daughter Merih Sara. I hope this work could be a humble compensation for their patience.

I would like to make a final remark: I want to thank again my advisor and all my dissertation committee members for their helpful comments, and I take full responsibility of errors and inconsistencies, if any, that may remain in this document.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Advancement in deep learning (DL) based biometric identification and the proliferation of affordable sensors made biometrics pivotal players in authentication and surveillance systems. For instance, major companies (e.g., MasterCard, AliPay) already adopted facial/voice-based authentication as part of their security measures. Furthermore, governments and private sectors use biometric recognition for a broader impact, such as identifying and catching "person of interest", targeted advertisement or border protection. While these technologies could have enormous impacts, existing biometric authentication and surveillance systems are vulnerable to several kinds of attacks, and also jeopardize the privacy of people's sensitive data. Although biometric-based systems offer superior usability and advantage for various use cases, they have to i) defend against different kinds of impersonation attacks and ii) protect the privacy of biometric data against adversaries.

This dissertation aims to provide solutions to above challenges. First, I will present vulnerabilities against impersonation attacks in an authentication setting. Our study shows that many cloud-based audio/visual recognition systems (e.g., Amazon Rekognition) can be defeated by the crudest impersonations. Then, I will present our live biometric verification system, the Real Time Captcha (rtCaptcha), a practical approach that places a formidable computational burden on the attacker by combining dynamic, live detection with a randomized Captcha challenge for stronger security. Second, I will present our privacy-preserving remote biometric authentication system, Justitia, which makes DL-inferences of biometric data compatible with the standard privacy-preserving primitives, like fuzzy extractors. Justitia lets a remote server to authenticate a client without revealing the biometric data in cleartext in the process of enrollment and authentication. Finally, I will propose a fuzzy (labeled-) private set intersection (FLPSI) protocol for privacy-preserving biometric search. FLPSI is a secure computation protocol that allows a client to search a biometric data over a sensitive database without revealing the query and search results to the server.

CHAPTER 1
INTRODUCTION

Advances in machine learning have led to new technologies for accurately analyzing biometric data, e.g. facial and voice data. With the core accuracy challenge solved, while deploying biometrics-based systems, we now face two main problems: i) impersonation attacks and ii) privacy violations of biometric data. In the former, because of weak liveness detection mechanisms, users face with the risk of losing their credentials, accounts and sensitive data, that are protected by biometrics-based authentication systems, to an impostor. In the latter, attackers can compromise large biometric databases since conventional biometrics-based authentication and recognition systems store and process biometric data in the cleartext at the enrollment and authentication/identification times. Even though these technologies bring great usability and advantage to end-users, companies and law enforcement, risks and flawed applications raise many questions and are hindering their usage. In this proposal, I will introduce the risks, vulnerabilities and our solutions to them.

With the increasing popularity of facial/voice-based authentication systems, major companies, e.g., MasterCard [1], Uber [2] and AliPay [3], already adopted these systems as part of their security measures. In particular, users can now authenticate themselves to one of Machine Learning as a Service (MLaaS)-based online services by using their mobile phone to show themselves performing simple tasks like blinking or smiling in front of its built-in camera. However, recent work on modeling a person's face/voice (e.g. Face2Face [4], DeepFake [5]) allows an adversary to create very authentic video/audio of any target victim to impersonate that target. Furthermore, our study shows that many of the publicly available MLaaS for facial/voice recognition (e.g. Microsoft Cognitive Services [6], Face++ [7] or Amazon Rekognition [8]) are vulnerable to even the most primitive attacks that fool these liveness checks and impersonate the target. All it takes to launch such attacks are a few pic-

tures and voice samples of a victim, which can all be obtained by either abusing the camera and microphone of the victim's phone, or through the victim's social media account. Other than forging an impostor biometric, adversaries may also authenticate themselves as the target by using replay and presentation attacks [9, 10, 11], building or re-enacting a 3D facial or video face [12, 13, 4, 14], or fooling the DL model through perturbed pixels [15]. To stop or mitigate these impersonation attempts, at least in the large scale, a biometric-based authentication system has to use an effective liveness verification technique, instead of playing the cat-and-mouse game between DL systems of the attack and defense sides.

This dissertation initially presents our Real Time Captcha (*rt*Captcha) system, which stops or slows down these attacks by turning the adversary's task from creating authentic video/audio of the target victim performing known authentication tasks (e.g., smile, blink) to figuring out what is the authentication task, which is encoded as a Captcha [16]. Specifically, when a user tries to authenticate using *rt*Captcha, they will be presented a Captcha and will be asked to take a "selfie" video while announcing the answer to the Captcha. As such, the security guarantee of our system comes from the strength of Captcha, and not how well we can distinguish real faces/voices from synthesized ones. Our evaluations show that, thanks to the humans' speed of solving Captchas, adversaries will have to solve Captchas in less than two seconds in order to appear live/human and defeat *rt*Captcha, which is not possible for the best settings on the attack side.

Once the large scale compromising attacks have been solved in the authentication setting, we can now focus on the second major problem of the biometrics-based systems: privacy-protection of biometric data. Conventional deployment of biometric authentication systems involves storing bio-templates in remote servers in cleartext. However, this raises serious privacy concerns as the biometric databases occasionally compromised. For instance, millions of fingerprint and facial biometric records have recently been stolen from the European and Indian databases [17, 18]. Current solutions propose keeping these templates on the client's device, outside the server's reach. However, this binds the client to

the initial device, which prevents recovery. A more attractive solution is to have the server authenticate the client, thereby decoupling them from the device.

Unfortunately, existing biometric template protection schemes either suffer from the practicality or accuracy. Existing privacy-preserving methods do not accommodate the state-of-the-art DL methods, as they are tailored to hand-crafted feature space of specific modalities in general.

In this dissertation, I secondly present our novel pipeline, Justitia [19], that makes DL-inferences of face and voice biometrics compatible with the standard privacy-preserving primitives, like fuzzy extractors (FE). To this end, we first form a bridge between Euclidean (or cosine) space of DL and Hamming space of FE, while maintaining the accuracy and privacy of underlying schemes. I also introduce efficient noise handling methods to keep the FE scheme practically applicable.

We implement an end-to-end prototype to evaluate our design, then show how to improve the security for sensitive authentications and usability for non-sensitive, day-to-day, authentications. Other than authentication functionality, Justitia is also a usable tool to prevent accessing sensitive credentials (like changing passwords), authorize social media posts, or bind public/private key to an identity.

According to our analysis, Justitia achieves the same, 0.33% false rejection at zero false acceptance, errors as the plaintext baseline does on the YouTube Faces benchmark. Moreover, combining face and voice achieves 1.32% false rejection at zero false acceptance. According to our systematical security assessments conducted through prior approaches and our novel method, Justitia achieves ˜25 bits and ˜33 bits of security guarantees for face- and face&voice-based pipelines, respectively. Further, our novel security assessment technique could be used as a black-box method in further biometrics research.

Some application scenarios may require querying a biometric data over a sensitive and private database in a privacy-preserving manner, which means that server should not know either of the query data and the search result. For instance, real-time identification of per-

sons in footage collected by surveillance equipment is a current practice that uses plaintext-based facial recognition systems. While real-time surveillance may be beneficial to public safety, there are serious objections due to privacy concerns [20, 21, 22]. Tracking "persons of interest" may be warranted, but tracking *everybody else* in the process (i.e., dragnet surveillance) is often unacceptable. In this context, a privacy-preserving biometric search plays a crucial role to leverage the achievements of DL systems without putting people's sensitive data into risks.

Even though Justitia achieves privacy protection for the client's biometric data for a single verification task, it does not scale for searching a biometric data over a large scale database. That is, parties have to repeat Justitia pipeline for each database entry to implement a privacy-preserving database search from Justitia. Instead, in this dissertation, I finally present a practical solution for private querying of a real-life biometric scan (e.g., a person's face) against a private biometric database. The querier learns only the label(s) of a matching scan(s) (e.g. a person's name), and the database server learns nothing. We formally define *Fuzzy* Labeled Private Set Intersection (FLPSI), a primitive computing the intersection of noisy input sets by considering closeness/similarity instead of equality. Our FLPSI protocol's communication is *sublinear* in database size and is concretely efficient. We implement it and apply it to facial search by integrating with our fine-tuned toolchain that maps face images into Hamming space.

We have extensively tested our system, achieving high performance with concretely small network usage: for a 10K-row database, the query response time over WAN (resp. fast LAN) is 146ms (resp. 47ms), transferring 12.1MB; offline precomputation (with no communication) time is 0.94s. FLPSI scales well: for a 1M-row database, online time is 1.66s (WAN) and 1.46s (fast LAN) with 40.8MB of data transfer in online phase and 37.5s in offline precomputation. This improves the state-of-the-art work by $9 - 25\times$ (on WAN) and $1.2 - 4\times$ (on fast LAN). FLPSI achieves a false non-matching rate is 0.75% for at most 10 false matches over 1M-row DB, which is comparable to underlying plaintext matching.

1.1 Contributions and Outline

This dissertation presents the following contributions to the literature.

Showing the insecurity of MLaaS. In Chapter 2, I first present how easy to fool existing cloud-based face/voice recognition services (e.g. Microsoft Cognitive Services [6], Face++ [7] or Amazon Rekognition [8]) even through the most primitive attacks. According to our analysis, using such services as part of an authentication pipeline makes the whole system vulnerable against simple impersonation attacks.

Preventing large scale compromising attacks. Even though researchers try to detect impersonation attacks by using machine learning techniques, the shelf-life of such mechanisms is not long enough since the attackers have access to same amount and quality of data and can always play and win this cat-and-mouse game. Instead, again in the same Chapter 2, I introduce our novel mechanism, rtCaptcha, that detects liveness of the client by binding her biometrics to response of a random challenge. This forces the attackers to involve the standard cryptographic challenge-response mechanism, and brings them an unavoidable computation burden that tell impostors and humans apart.

Privacy protection of biometrics in authentication setting. In Chapter 3, I present our privacy-preserving remote biometric authentication system, Justitia. I initially show how to address space incompatibility problem between deep learning, which handles the fuzzy biometric data mostly through Euclidean or cosine similarity space, and cryptographic primitives that achieves privacy preserving. While doing this I also show how to avoid accuracy loss of underlying deep learning techniques without sacrificing from the performance.

Measuring security of biometric authentication systems. One of the most challenging task in biometrics-based system is measuring the achieved security against smash-and-grab attacks, which assume the adversary can get the whole database and tries to infer the biometrics of any enrolled user in it. Despite the many attempts in the literature, there

is no universally agreed mechanism to measure the entropy in a biometrics-based system. In Chapter 3, I also present a novel black-box security assessment technique that assumes a powerful threat model, where the adversary can leverage from the domain knowledge and device a generative adversarial network-based model to create photo-realistic impostor samples to match an enrollment in the database.

Privacy protection of biometrics in recognition setting. In Chapter 4, I present fuzzy labeled private set intersection (FLPSI) protocol, which is designed for protecting privacy of fuzzy data (such as biometric data) in a large scale database search. I initially show how to map biometric inputs of a person into a *t-out-of-T* exact matching set elements without a major accuracy loss. Then, I present how to build an LPSI protocol upon AES blockcipher, garbled circuits, latice-based fully homomorphic encryption and secure secret sharing algorithms to fundamentally solve the privacy problem of surveillance data in a semi-honest two-party computation model. That is how a client (e.g., police officer, retail store etc.) can search the biometrics of regular people over a server's sensitive database without sacrificing from the real-time performance and accuracy loss compared to the plaintext systems.

Formal security definition and proof of biometrics-based cryptographic protocols. In the same chapter, I also present a novel simulation-based security definition technique for the protocols that accept fuzzy inputs, and then formally prove the security of FLPSI protocol. In secure multi-party computation (MPC), the preferred simulation-based security definitions offer composable guarantees with a requirement of a precise specification in ideal-world behavior, which is not known how to achieve for fuzzy functions. That is why the current practice is using game-based definitions that are not composable, but allow to bound instead of precisely specifying the adversary success. Our novel definitional approach allows the best of both worlds: a composable ideal-real world simulation, and yet bound adversary success rather than exactly specifying it. This is a generic definition and incorporates optional leakage, and thus can serve as a template in defining primitives in the biometric space for the future research.

CHAPTER 2

PREVENTING LARGE SCALE COMPROMISING ATTACKS

In this chapter, I will present *rt*Captcha, a real time Captcha based liveness detection system, to mitigate large scale impersonation attempts in face- and voice-based remote biometric authentication settings.

2.1 Introduction

With automatic facial/voice recognition becoming more accurate[1] and available[2], online services are increasingly allowing their users to authenticate using their face/voice. In such authentication schemes, all one needs to do to authenticate is to perform simple tasks like smile, blink or nod in front of his/her mobile phone, while the phone's camera will record the video of the user performing such a task and send it to the service provider. The authentication is successful if the service provider determines that the received video is indeed that of the expected user performing the required task.

While these new generation facial-recognition-based authentication systems offer superior usability and are robust in benign settings, existing works already have shown that they are quite vulnerable to several kinds of attacks. Depending on how the malicious video/media is fed into the authentication system in order to impersonate a user, existing attacks can be classified as *presentation attacks* and *compromising attacks* (see Fig. 2.1 for an illustration).

As we can see in Fig. 2.1, presentation attacks physically "present" the impersonating media (mostly static photos or masks) to the sensors used by the authentication system, while compromising attacks involve tempering/fabricating the digital output of the sensors.

[1] Facebook [23] and Google [24] have respectively achieved recognition accuracies of 97.35% and 99.63% on faces under different illumination, pose and facial expressions

[2] Through cloud services like Microsoft Cognitive Services or Amazon Rekognition

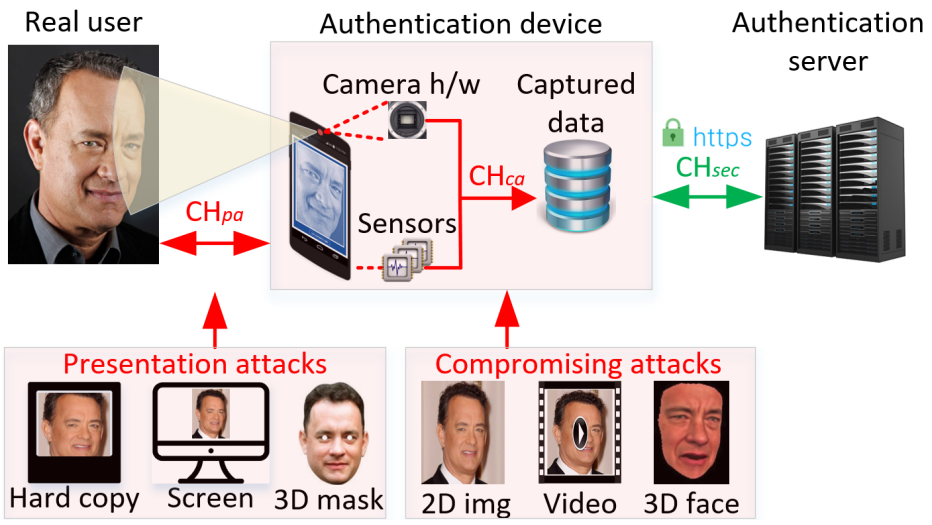


Figure 2.1: Attack channels specified by ISO/IEC 3017-1 standard and possible spoofing media types deployed via these channels. CH_{pa} and CH_{ca} represent presentation and compromising attack channels respectively.

An observation we can make here is that compromising attacks are potentially much more scalable than presentation attacks, because compromising attacks can happen entirely in the digital world, while presentation attacks are restricted to physically presenting something to the target system. Recent advances in face modeling (e.g. Face2Face [4]) further make it easier to automate compromising attacks. In particular, all an attacker needs to generate authentic looking video of the target victim performing the task necessary for authentication are pictures or videos of the victim. Such materials can be obtained either through abusing the camera of the authentication device, or through accessing publicly available data from the victim's social media accounts– all very doable once the attacker has a foothold on the authentication device (e.g. have their malicious app installed and granted the right permissions). Once such "authentication video" is generated, the attacker will need a way to feed it into the authentication system to launch a compromising attack. This final step can be achieved in a number of ways, e.g. through compromising the authentication device's OS (and then modifying the output buffer of the camera) or through reverse engineering the authentication protocol and directly talking to the server.

Based on the above observation that compromising attacks are much easier to launch

in a large scale than presentation attacks, we will focus on the former in this paper. However, as we will note in Sect. 2.8, our solution is also expected to make state-of-the-art presentation attacks more detectable.

Regarding existing work in defense against both presentation and compromising attacks, we can argue that *most defenses focus on making it harder for the adversary to generate sufficiently good video of the victim performing the "authentication task."* An obvious example is to perform better analysis of the received video to determine if it came from a real human or was somehow synthesized. A less obvious example is liveness detection, which is mainly used to defeat "replay" attacks that employ static pictures of the victim; and this simply pushes the attackers to create animated 3D models, instead of continuing to use static pictures. *What remains constant is that the "authentication task" is predefined, fixed and known to the adversary, and this allows the adversary to develop new technologies to create authentic videos of the victims performing the right task.*

In this work, we propose a different, orthogonal approach; instead of making it harder for the adversary to generate video of the victim performing the known task to fool our system, *we make it harder for the adversary to know what is the required task the user must perform in order to successfully complete the authentication*. In particular, at each authentication attempt, instead of asking the user to perform simple actions such as blink or smile in front of the camera on the authentication device, our server will send the user a Captcha and have the authentication device take a video of the user answering that Captcha. *For a normal user, this will be easy, but for an automated attack, this will mean automatically solving the Captcha before feeding the answer into the algorithm for generating fake video of the user answering the Captcha.* As in all Captcha schemes, we can assume that the time it takes for a real user to solve the Captcha is significantly shorter than it is for the adversary (even if some human intervention is involved, see Sect. 2.7 for our evaluation of this claim), and only authenticate in case the correct response is received within some threshold time.

To summarize, the high level idea of our system is to turn the attacker's task from generating good quality video of the victim to one of breaking Captcha– a task that is well understood and studied by the security community. *A point worth noting is that while the security of our system depends on the security of the Captcha scheme we used, we consider the security of Captcha an orthogonal problem. We believe Captcha (i.e. being able to tell if a user of an online service is a human) is so important, not only to us, but to the whole online ecosystem, that the security of Captcha will continue to improve, and any improvement in the security/robustness of Captcha can be easily adopted to our system.*

We have implemented our high level idea in a prototype system called *rt*Captcha. Our user study shows that normal human response time to the Captcha presented at authentication time is less than 1 second even for the most complex scheme. We also conducted experiments on the same challenges with existing Captcha solving services and state-of-the art techniques which has 34.38% average recognizing accuracy and 6.22 seconds average execution time [25]. In other words, there is a very large safety margin between the response time of a human solving a Captcha and a machine trying to break one.

In summary, the contributions of this work are:

1. We perform an empirical spoofing analysis on current cloud based audio/visual recognition and verification systems that use state-of-the art data-driven deep learning architectures.

2. We propose a practical and usable liveness detection scheme by using security infrastructure of Captchas to defeat even the most scalable and automated attacks.

3. We perform analysis on existing automated and man-powered Captcha-breaking services and state-of-the art Captcha-solving algorithms by using the most popular Captcha schemes in the market.

4. We have implemented a prototype Android application and conducted a user study.

5. Our evaluations show that audio response of a normal human being to a Captcha challenge is much shorter than automated attacks which have state-of-the art synthesizers and Captcha-breaking methods.

In the rest of the paper, we will introduce our threat model in Sect. 2.2, present existing attacks and defense mechanisms in Sect. 2.3, present our experiment on launching compromising attacks against existing, publicly available systems in Sect. 2.4, describe design and details of *rt*Captcha in Sect. 2.5, evaluate the human performance and existing Captcha breaking tools on solving Captcha challenges along with usability and user acceptance of *rt*Captcha in Sect. 2.6, provide the security analysis of *rt*Captcha against our threat model in Sect. 2.7, discuss about limitations and future works in Sect. 2.8, and conclude in Sect. 2.9.

2.2 Threat Model

In this work, we focus on defending against *powerful, automated* compromising attacks. We assume the following threat model:

- the authentication device is a mobile phone with a camera and a microphone,
- the authentication server is *not* compromised,
- the kernel of the authentication device can be compromised,
- there is no form of attestation mechanism on the authentication device, since software attestation is theoretically security by obscurity and hardware attestation is not yet available for phones,
- the protocol between the client app running on the authentication device and the server is known to the attacker, thus the attacker can run malicious version of the client app on the authentication device that will completely control the camera and microphone input to the authentication server,

- the attacker can abuse the camera and microphone on the authentication device to collect samples of the face and voice of the victim; the collected samples can then be used to generate models of the victim's voice and face, which can then be used to synthesize videos and audios for impersonating the victim during a future authentication session,
- the attack needs to be completely automated, and it needs to happen on the victim's authentication device, otherwise we believe the attack cannot scale.

Out-of-scope: Based on above assumptions, we consider the following attacks out of scope for this work. 1) Presentation attacks which involve showing 2/3D printed and wearable masks, hard copy photos or device screens displaying the target's face and other rudimentary manipulations, (since this means the attack must happen on an authentication device physically controlled by the attacker, thus violating the last assumption in the threat model). 2) Facial mimicry manipulations through face reenactment [26, 4], where facial expressions of an imposter solving the Captcha are captured and applied in real time to a 3D model of the victim to synthesize the victim's face responding to our liveness detection challenge[3] (this requires a source actor to perform the act before the software can map it to the target subject, which is not a practical and scalable attack scenario for automated tasks).

2.3 Related Work

In this section, we summarize existing liveness detection methods against both presentation and compromising attacks.

2.3.1 Presentation Attacks and Defenses

The requirement of liveness detection systems against face spoofing attacks was first introduced by researchers who showed that existing face authentication applications for both desktop and mobile platforms are vulnerable to single image spoofing [9, 10]. As a defense

[3] In [4], this can be achieved with a 20ms latency.

mechanism against this attack, researchers proposed challenge-response based liveness detection mechanisms that involve user interaction such as smile, blink, lip or head movement, etc. [27, 28]. However, frame switching or video-based attacks proved how easy it was to bypass smile or blink detection since they have arbitrary facial frames creating a motion to fulfill desired challenges [11]. Both image and video-based attacks are deployed as presentation attacks, but, they also are suitable for a compromising attack scenario. However, the latter attacks and corresponding defense mechanisms have been sophisticated for either presentation or compromising attacks.

Against presentation attacks, researchers mainly focused on discriminating 3D structure, texture or reflectance of a human face from a planar surface. To this end, 3D shape inferring features such as optical flow and focal length analysis, color and micro texture analysis, or features extracting reflectance details (such as visual rhythm analysis) have been proposed against presentation attacks [29, 30, 31, 32, 33, 34]. On the other hand, researchers proposed a wearable 3D mask for presentation attacks to defeat all of these anti-spoofing methods. However, reflectance and texture analysis-based defense mechanisms have also been proposed against 3D mask attacks [35, 36, 37, 38]. It is worth noting that many different approaches and design choices have been proposed at the competitions on the countermeasures to presentation attacks [39, 40, 41].

2.3.2 Compromising Attacks and Defenses

Recent advances in 3D face model creation (using a couple of images) have been employed to launch compromising attacks [13]. In order to capture enough raw material for model generation, the victim's face/voice could be captured through a user interface (UI) redressing attack caused by a malicious app that allows particular permissions (e.g. draw-on-top on Android device [42]) without his/her notice. To generate a 3D face model from captured image/video, the most suitable approach in existing literature is to use pre-built 3D Morphable Models (3DMMs) [43, 44, 45]. 3DMMs are the statistical 3D representations

built on facial textures and shapes of many different subjects (e.g. 10,000 faces in [44]) by incorporating with their facial expressions and physical attributes at the same time. Once built, a 3DMM is ready for reconstruction according to facial attributes of a victim's face. The details of building a 3D face model could be found in [44], but the overall pipeline is as follows. First, facial landmarks which express pose, shape and expression are extracted from the victim's face. Then, the 3DMM is reconstructed to match the landmarks from both the 3D model and the face. Hence, pose, shape and expression of the face are transferred to the 3DMM. After reshaping the 3DMM, texture of the victim's face is conveyed to the 3D model. Since a 2D face photo/frame does not contain full representation of its 3D correspondence, a photo-realistic facial texture is generated from the visible face area in the photo/frame for missing parts in the 3D representation [14]. Then, this 3D face is transferred into a VR environment to fulfill requested challenge tasks (e.g., smile, blink, rotate head, etc.).

On the defense against compromising attacks, even though some inertial sensor-assisted methods increase the security of facial authentication systems [46], such a compromised environment with given permissions can allow attackers to use additional sensor data to manipulate the motion of 3D face model in a VR environment. Another defense mechanism against these attacks, especially against VR based ones, could be analyzing the authentication media by using forensic techniques to detect forged audio/video [47, 48]. However, since 3D face models are created from scratch with high fidelity texture data, these methods could not detect any forgery on spoofing media. On the other hand, new approaches (such as discrepancy analysis on color filter array of camera sensor noise or multi-fractal and regression analysis on discriminating natural and computer generated images) could be used as countermeasures against 3D-face-model-based attacks [49, 50]. However, attackers can extract genuine noise patterns or features from existing or captured images to embed them into generated video in a compromised device, thus, these defense mechanisms also fail against our threat model [51]. Hence, defense mechanisms against compromised attacks

should not rely on additional device data as suggested in previous works.

User authentication through audio response to text challenges is first proposed by Gao et al. [52]. However, their goal is mainly to distinguish between natural and synthesized voice. Their results show that human responses can pass the system with 97% accuracy in 7.8 seconds average time while a very basic text-to-speech (TTS) tool (Microsoft SDK 5.1[4]) can pass the system with 4% success rate. In contrast to *rt*Captcha, the work in [52] use plain-text challenges and thus allows the attacker to easily learn what is the task involved in the liveness detection challenge, and thus can be easily defeated by more sophisticated real-time synthesis of the victim's voice (e.g. [53, 54]). Shirali et al. [55] proposed a scheme that involves audio Captchas. In their system, challenges are sent to users in audio formats and users give audio responses back to the system. They use audio features such as Mel-Frequency Cepstral Spectrum (MFCC) to correlate challenge and response audios at the decision side. They achieved 80% of authentication accuracy on average. However, since breaking audio Captchas are as easy as breaking plain-text challenge by using a speech-to-text application, this work also does not provide good defense against compromising attacks. To the best of our knowledge, *rt*Captcha is the first approach that binds a text-based Captcha challenge response with user's biometric data in the realm of audio/visual liveness detection.

2.4 Evaluating the Security of Existing Systems

In this section, we study how vulnerable the most popular facial and voice-authentication systems are to the compromising attacks which motivate our work in *rt*Captcha. In particular, we tested all studied systems against compromising attacks of various levels of sophistication in terms of how they create the impersonating video/audio of the victims, using open source spoofing datasets.

[4] www.microsoft.com/en-us/download/details.aspx?id=10121

Table 2.1: Baseline and Spoofing results of cloud based face authentication systems

Cognitive Service	Baseline / Conf. (%)		Number of Verified Faces Over 10 / Overall Confidence Rate (%)					
	TP	**TN**	$3D_{sf}$	$3D_{fg}$	$3D_{ct8}$	$2D_{car}$	$2D_{ske}$	$2D_{fem}$
Microsoft	**10 / 78**	**10 / 65**	10 / 70	10 / 75	10 / 70	10 / 82	10 / 84	10 / 86
Kairos	**10 / 80**	**8 / 58**	10 / 75	10 / 78	10 / 73	10 / 91	10 / 83	10 / 80
Face++	**10 / 87**	**10 / 83**	10 / 86	10 / 71	10 / 72	9 / 77	7 / 80	7 / 75
Amazon	**10 / 97**	**10 / 82**	10 / 89	8 / 77	9 / 67	7 / 84	6 / 84	9 / 89

2.4.1 Facial Authentication Systems

We tested most popular cloud-based facial recognition services that are provided or funded by major companies such as Microsoft [6], Amazon [8], AliPay (Face++) [7] and Kairos [56].

Database: We tested each studied system against videos showing real/fake faces. We used subjects from the open source CASIA Face Anti-Spoofing Database [57]. In particular, we took the genuine videos from the CASIA Face Anti-Spoofing Database and: 1) used them as positive samples to test each studied system, and 2) used them as samples for generating synthesized videos, and used them as negative samples against each tested system. For our experiment, we used the first 10 subjects from the CASIA database.

Synthesizing methods: We tested each studied system against videos synthesized using methods with various levels of sophistication. Fig. 2.2 presents a complete set of synthesized video for a user in the database (5^{th} subject in training set). We can summarize the synthesizing techniques employed starting from the most complex to the simplest as followed:

1. 3D Face Model: This is the state-of-the art method for generating fake face video for the purpose of compromising attacks [13]. For our experiments, we generated 3D face models from genuine videos of each subject in our dataset by using three different tools: i) Surrey Face Model (labeled as $3D_{sf}$ in Fig. 2.2), a multi-resolution 3DMM and accompanying open-source tool [45]; ii) FaceGen[5] ($3D_{fg}$), and iii) demo version of

[5]https://facegen.com

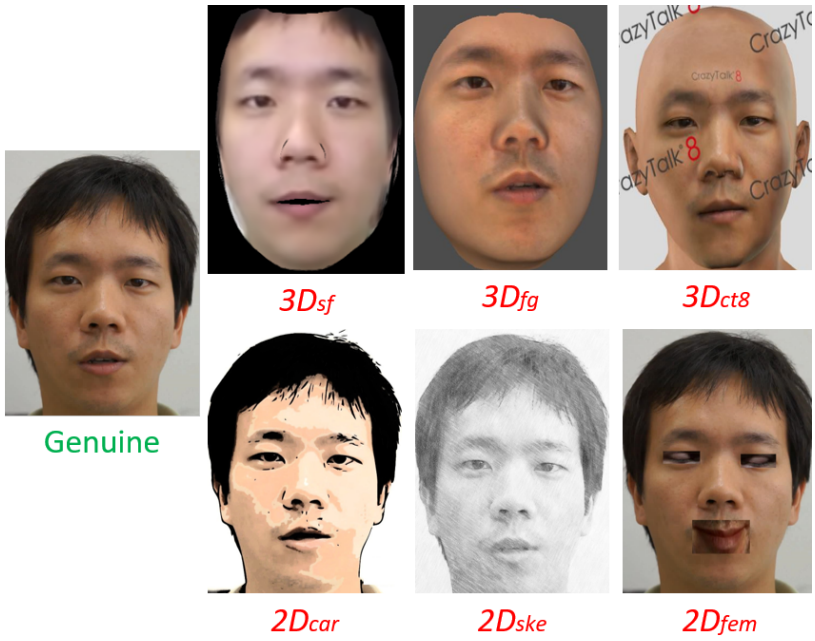


Figure 2.2: A full media set including genuine and fake versions of it for a subject in our face authentication database.

CrazyTalk8[6] ($3D_{ct8}$) commercial tools used for 3D printing or rendering 3D animation and game characters. Although the demo tool puts a brand mark on 3D models, they don't seem to have any effect on the effectiveness of the attack.

2. Cartoonized and Sketch Photos: To detect whether the face authentication systems check the texture information or not, we convert randomly grabbed frames from the genuine videos to cartoonized and sketch forms[7]. We express these manipulations with $2D_{car}$ and $2D_{ske}$, respectively.

3. Fake Eyes/Mouth Photo: Finally, we replaced eyes and mouth regions of the stationary photos with fake ones which are cropped from an animation character. We conduct this attack method to prove that facial authentication and verification systems only focus on the location of facial attributes. To create appropriate fake eyes and mouth, we first extract the facial landmarks to get their regions. Afterwards, we reshape our fake eyes and mouth templates to exactly fit their corresponding regions. This manipulation is

[6]https://www.reallusion.com/crazytalk
[7]http://www.cartoonize.net

represented by $2D_{fem}$ in the evaluation results.

Methodology: We experimented with each studied service as followed: we enrolled each subject with his genuine face sample. After the enrollment, we established the baseline performance of each service by presenting one genuine face from the enrolled user (thus measuring its true positive, TP) and one genuine face from a different user (thus measuring its true negative, TN). To test the robustness of each service against attacks, we presented each service with all of our synthesized videos. To make the experiment more realistic, we generated the synthesized videos using samples different from those used for registration. The success rate of each synthesis technique and its overall similarity rates (which is the tested service's measure of how close the presented video is to the one from registration) are in Table 2.1. Since most of the services accept 50% of similarity rate for correct verification, we also consider this threshold in our experiments.

Findings: First of all, under benign conditions, we find the analyzed services have an overall baseline true positive (TP) of 100% and an overall baseline true negative (TN) of 95%, with 85.5% and 75.7% overall confidence rates, respectively. Our results also show that Amazon Rekognition service performs best among all tested service, since its confidence rates on both TP and TN are highest. Unfortunately, we also find that all of the analyzed services are vulnerable against almost all of the tested synthesis techniques. Results show that 92.5% of the spoofed faces are detected as genuine copies with an average similarity rate of 79%. More specifically, Cartoonized and Sketch photo attacks showed that the texture information is not considered in the authentication process at these systems. When we made detailed analysis to understand the reason for a lower matching rate in the Sketch photo attack, we conclude that it is because the tested services cannot detect facial region on those samples. The success of Cartoonized and Sketch photo attacks highlights that attackers can succeed without putting much effort in building a high fidelity facial texture; it would add to the latency in generating the synthesized video to answer the liveness detection challenge presented. Moreover, results of fake eyes/mouth spoofing

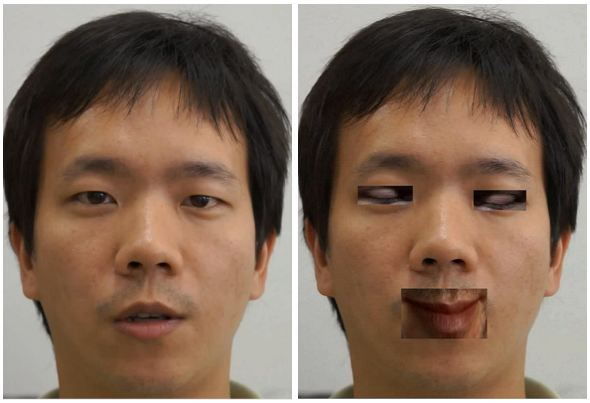


Figure 2.3: Smiling probabilities of genuine and fake samples with 78.52% similarity rate.

amusingly proved that all of these systems are only using the landmark locations as the facial feature set on their face authentication protocol. 3D facial model spoofing results also support these outcomes since we used non-sophisticated tools to create 3D models and facial textures. Even though the demo software puts some brand marks over the generated face, we still get very high similarity rates with these 3D models. Hence, faces created by one of the latest 3D facial model generation software (e.g. [12, 14]) are very unlikely to be detected as fake by these services. As a result, we can make an inference that even if a facial authentication scheme uses a challenge-response based liveness detection mechanism (such as smile/blink detection) accompanying one of these services, it will be very easy to spoof such a scheme even by conducting a rough switching frame manipulation (e.g. when asked to blink, go from a frame with open eyes to one with closed eyes for a short time) or using a demo application to create 3D face model and manipulate the model to answer the challenge. For instance, Fig. 2.3 shows how easy it is to get a high smiling probability from Microsoft Cognitive Service even with a rough manipulation on a genuine face while preserving similarity rate around 78.52%. Assuming a security mechanism that uses smile-detection as a liveness clue and MS Face API to authenticate user face (as Uber does for driver authentication), then a crude attack as in the figure can defeat this mechanism without using any sophisticated tool or algorithm.

2.4.2 Voice Authentication Systems

In this section, we show that automatic speaker verification (ASV) systems also have similar vulnerabilities to compromising attacks as do their facial recognition counterparts. To make a clear demonstration, we systematically conducted attacks with synthesized voices from the MS Speaker Identification (SI) service by using open-sourced synthesized speech data sets.

Database: In our experiments, we used two different datasets, from ASV Spoofing Challenge [58] (V_{asv}) and DNN based speaker adaptation work by Wu et al. [54] (V_{dnn}). The first dataset, V_{asv}, contains both genuine and synthesized voice samples for a total of 106 male and female users. ASV Spoofing Challenge is organized for Interspeech 2015 Conference to determine best technique detecting spoofing methods against automatic speaker verification systems. Hence, organizers published V_{asv} dataset containing synthesized versions of genuine data which are generated by 7 voice conversion (VC) and 3 speech synthesizing (SS) techniques. We denoted these samples from V_{asv}^{1} to V_{asv}^{10}. Some of the synthesized data have published before the submissions evaluated, which are called *known attacks*, and some of them are used only for evaluation which are called *unknown attacks*. Overall spoofing detection accuracy of the submissions is around 97% for *known attacks* and 91% for *unknown attacks*. It is worth noting that almost all submitted countermeasure methods perform worst on the last SS based samples (V_{asv}^{10}).

The second dataset, V_{dnn}, contains both genuine and synthesized samples for one female and one male speaker, where the synthesized speech samples were generated by using 7 different settings of their DNN framework. These samples are denoted as V_{dnn}^{1-7}. Objective and subjective experiments prove that all DNN techniques in the paper have better adaptation performance than the hidden Markov model baseline in terms of naturalness and speaker similarity. Hence, we use this dataset as the state-of-the art spoofing attacks.

Methodology: We first enrolled 10 users using their genuine samples from the two datasets, (2 users from V_{dnn} and 8 randomly selected users from V_{asv}), each with a total

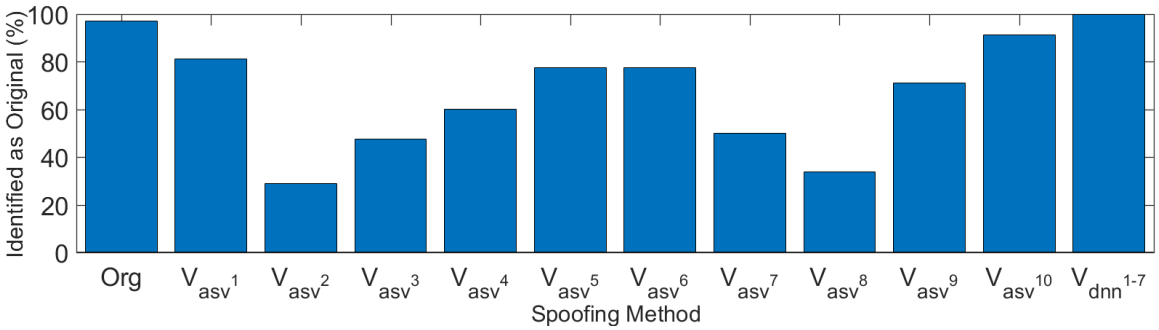


Figure 2.4: Success rate of speaker spoofing attacks to Microsoft SI service.

of 30 seconds of speech samples. We then tested the targeted service against 10 genuine samples from the enrolled user, as well as 7 (for V_{dnn}) or 10 (for V_{asv}) synthesized samples generated for the enrolled user by each tested technique, and evaluated if each tested sample was successfully identified as the enrolled user.

Findings: In Fig. 2.4, we present the genuine identification results for the genuine samples (Org), synthesized samples generated by 10 different methods in the V_{asv} dataset (V^1_{asv} to V^{10}_{asv}) and 7 different DNN methods in the V_{dnn} dataset from left to right. V^{1-7}_{dnn} average result is given for 7 DNN based synthesizers in the V_{dnn} dataset. First of all, we note that 97% of the genuine samples were identified correctly. Hence, it shows that the cloud service is working accurately for the recognition tasks. On the other hand, samples synthesized by various tested SS and VC methods have an average success rate of 64.6%. More specifically, even with the worst performing VC tool, there are still 28.75% of the synthesized samples identified to be from the real enrolled user. Additionally, samples from open sourced TTS synthesizers (10^{th} method of V_{asv}) can have a 90% chance of being considered legitimate. Finally, if an adversary generate synthesized voice of a victim by using a DNN based approach, the SI service identify the forged speakers as a genuine one 100% of time (this is true for all methods/settings in V_{dnn}). The results also prove that the parameter space to synthesize is much more bigger than those which are used by verification methods. That is why, even the simplest VC approach can tune the voice characteristics of the victim to the level of a verification system's requirements.

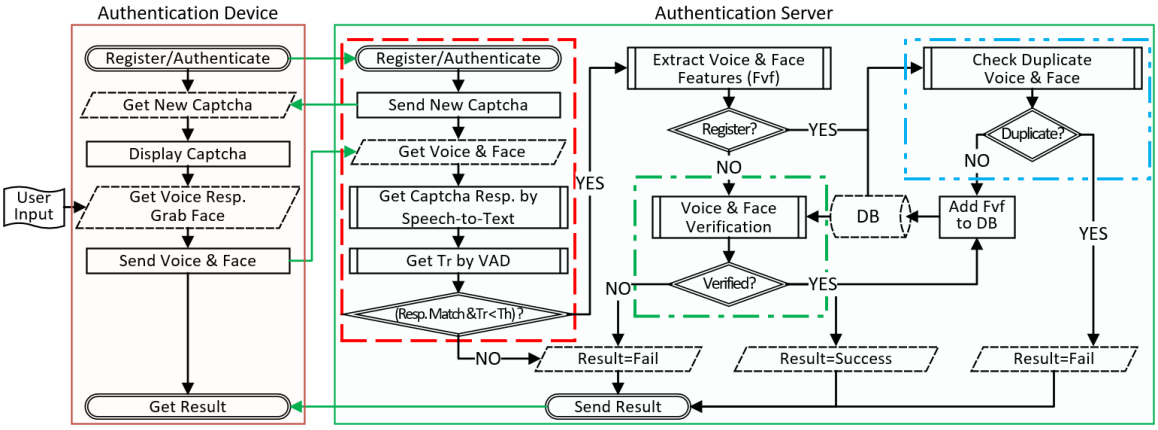


Figure 2.5: Process flow diagram of *rt*Captcha. T_r, T_h and F_{vf} refer to user response time, human response time threshold and face & voice feature vector, respectively.

2.5 Our Approach

The workflow of *rt*Captcha is summarized in Fig. 2.5. The workflow starts when an authentication device (i.e. mobile phone) needs to start an authentication/registration session; to proceed, it will establish a secure connection with our authentication server. Upon receiving requests over the secure channel, our server will generate and send a Captcha challenge to the authentication device and measure the time until the authentication device responds. The session will time out if no response is received after a predefined period.

On the authentication device, once it receives the Captcha challenge, it will display the challenge to the user and start recording the user's audio response. The client app running on the authentication device will also take a number of snapshots of the user at random time while he/she is responding to the challenge, using the front camera on the phone. We use the phone's voice recognition system to determine when the user has finished responding to the Captcha challenge; the captured voice and face samples will then be sent to the server using the established secure channel. To avoid unnecessarily utilizing the more expensive voice/facial recognition service, our server will perform initial sanity check of the response by transcribing the audio response received using a standard speech-to-text (STT) algorithm to determine if the response corresponds to the Captcha solution to the challenge we

sent. We will also determine how much time it takes for the user to start responding to the challenge by measuring when did the first speech activity happen in the received response. If the user took too long to start responding, we will consider the liveness test a failure and reject the authentication/registration request. If the received response passes the preliminary checks, we'll perform the more expensive analysis to determine if the validity of the received voice and face samples (the exact process will depend on whether the request is for authentication or registration, and we will detail the process for each case below).

Registration: Our analysis for registration is very simple and mostly involves sanity check of the received face and voice sample to make sure they came from a real human being to further avoid bot registration and avoid wasting resources to establish accounts for non-existent/non-human users. If necessary, we can also match the received samples against that of all existing users to detect attempts to register multiple accounts for the same person. If the face and voice samples pass all our tests, we will proceed to create the new user account and tie the received face and voice sample to that user and use them for future authentication sessions.

Authentication: For authentication requests, if the user is trying to authenticate as user X, we will compare the received facial and voice samples against the samples received at the establishment of account X. If the samples are verified as coming from user X, we can confirm the liveness and authenticity of the request; liveness is confirmed since the Captcha challenge is correctly answered, authenticity is confirmed through the matching face and voice sample. Thus, we will report to the user that the authentication is successful and let him/her log in as user X. Upon successful authentication of a user, we can also add the received face and voice sample for this authentication attempt to the user's record to improve his/her face and voice profile for future authentication.

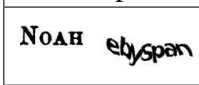
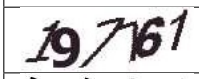
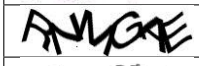
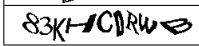
Using this framework, we can prevent the adversary from launching automatic, large-scale user impersonation using compromised phones. In the following, we will provide implementation details of the *rt*Captcha system.

2.5.1 Captcha Challenge

For our implementation of *rt*Captcha, we've employed a number of commonly used Captcha generation tools so we can experiment with and fine tune the difficulty level of our liveness detection. In the following we will give a brief description of the Captcha schemes we've experimented with.

In the literature, text-based Captchas are classified into three different categories according to font styles and positional relationships between adjacent characters. The three categories are, namely, character isolated (CI) schemes, hollow character schemes, and crowding characters together (CCT) schemes [25]. Some Captcha providers also use variable character sizes and rotations or different kinds of distortions and background noises to make their Captcha harder to break. For our experiments, we obtained Captcha samples used by Gao et al. [25] (which conducted generic Captcha breaking attacks on them). We also have modified the *Cool PHP Captcha*[8] framework to create variable size Captchas of short phrases or numbers to include random lines on background. Table 2.2 summarizes different Captcha schemes we have experimented with in our user study and evaluations.

Table 2.2: The most popular Captcha schemes

Sample	Scheme	Websites
	reCAPTCHA (CCT scheme)	linkedin, facebook, google, youtube, twitter, blogspot, wordpress
	Ebay (CCT scheme)	ebay.com
	Amazon (CCT scheme)	amazon.com
	Yandex (Hollow scheme)	yandex.com
	Yahoo! (Hollow scheme)	yahoo.com
	Microsoft (CI scheme)	live.com, bing.com

Regarding the hardness of these Captcha schemes, Brodic et al. [59] shows that an average Internet user can solve text and numeric Captchas in hollow and CCT (reCaptcha)

[8]Cool PHP Captcha is used in the reCaptcha scheme, and is available at https://github.com/josecl/cool-php-captcha

schemes at around 20 seconds on average (3 secs. min.). They also show that Captcha solving time is correlated with education and age. However, previous findings focus on the scenario where the user has to *type* in the answer to the Captcha, while in our case, *they only have to speak out the answer*, which should be faster and easier than typing. Thus, we have performed our own user study to determine how long it will take users to complete the liveness challenge in our settings. Our findings are reported in Sect. 2.6.

2.5.2 Transcribing Captcha Responses

As a first step in our validation of the face and voice samples received for the liveness test under *rt*Captcha, we transcribe the voice sample using a speech-to-text (STT) algorithm to see if it's a correct answer to the Captcha we sent. In our framework, we used a Hidden Markov Model (HMM)-based approach with a pre-trained dictionary. We used the open-sourced CMU Pocketsphinx library, Carnegie Mellon University's Sphinx speech recognition system [60], in our user study app since it provides a lightweight library working on mobile devices. CMU Sphinx is the state-of-the art solution among HMM based approaches. There also are many sophisticated alternatives for this step. For example, recently Baidu's open source framework Deep Speech 2 exceeds the accuracy of human beings on several benchmarks [61]. They trained a deep neural network (DNN) system with 11,940 hours of English speech samples. Cloud-based cognitive services such as Microsoft Bing Speech API[9] or IBM Watson Speech to Text[10] also could be used as STT algorithm for this step. However, network latency caused by audio sample transmission could be a drawback in our framework.

2.5.3 Audio Response Validation

Given a verified audio response of the Captcha challenge, the next verification process tests user response time to the challenge. Unlike typing-based responses, our further analysis

[9] https://azure.microsoft.com/en-us/services/cognitive-services/speech/
[10] https://www.ibm.com/watson/services/speech-to-text/

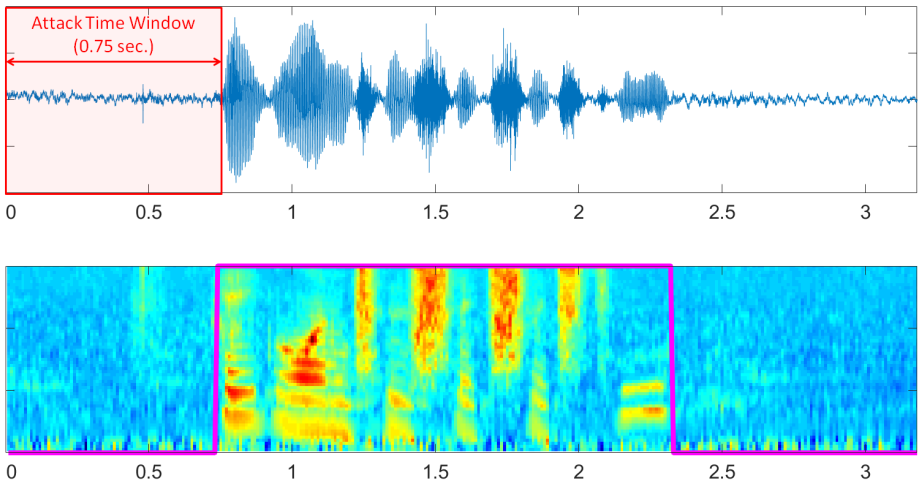


Figure 2.6: Time window for adversarial actions.

will show that giving audible response is much faster. Furthermore, the attacker's time window for breaking Captcha challenges and synthesizing a victim's face and challenge announcing voice is smaller than even the duration of audible response. As depicted in the top of the Fig. 2.6, adversarial action time is limited with the beginning of the speech activity.

Speech activity detection, also referred to as voice activity detection (VAD), is a ubiquitous method that has been studied and discussed in different contexts such as audio coding, content analysis and information retrieval, speech transmission, automatic segmentation and speech recognition, especially in the noisy environments [62, 63]. In our framework, we used a hybrid model that follows a data-driven approach by exploiting different speech-related characteristics such as spectral shape, spectro-temporal modulations, periodicity structure and long-term spectral variability profiles [64]. After getting different streams representing each of these profiles, the information from the streams is applied to the input layer of a Multilayer Perceptron classifier. The overall equal error rate of this approach is around 2% when a classifier is built with 30 hours data and tested on 300 hours data. Since our audio responses will be a few seconds, the error rate will be a few milliseconds. On the bottom of Fig. 2.6, we presented speech activity detection on the spectrogram of a sample Captcha response audio from our experiments.

Once user response time has been extracted, if it is within an expected human response time and not-longer than the breaking time of the corresponding Captcha scheme, we verify the challenge response as a genuine attempt. The reference response time window could be adapted for each user and Captcha scheme with his/her response times from the successful attempts since Captcha reading behavior could vary for each person and scheme.

2.5.4 Facial and Voice Verification

After getting a correct Captcha response within a real human response time, we verify the user's facial and voice data by using data from the registration phase. If the attempt is a new user registration, we again make facial and speaker recognition to check that the new user is not a duplicate one. Facial and speaker recognition and verification literature could be investigated under two different categories; first one is feature or descriptor-based (old fashioned) and the second one is a data-driven DNN-based (modern) approach. In our experiments, we used cognitive services of Face++ and Microsoft to verify a user's face and voice, respectively.

2.6 Evaluation

In this section, we present the results of our evaluation on *rt*Captcha to show that it provides a strong, yet usable, liveness detection to protect facial/voice-based authentication systems against compromising attacks. In particular, we have performed a user study to measure: 1) the time difference between a real user solving the Captcha presented by *rt*Captcha versus the time it takes for an algorithm to break it, 2) the usability and user acceptance of *rt*Captcha.

Note that our user study has been approved by the Institutional Reviews Board of Georgia Institute of Technology.

Table 2.3: Response times and challenge recognition accuracy of our approach (Human$_{aud}$), men-powered Captcha solving service (Attack$_{typ}$), OCR based (Attack$_{ocr}$) and state-of-the art Captcha breaking algorithms (Attack$_{best}$) [25].

Task	Captcha scheme	Time (secs) / Recognition accuracy (%)			
		Human$_{aud}$	Attack$_{typ}$	Attack$_{ocr}$	Attack$_{best}$
1	Plaintext	**0.77 / 91.9**	N/A	N/A	N/A
2	reCaptcha$_{num}$	**0.90 / 87.1**	22.11 / 96.7	2.98 / 0	10.27 / 77.2
2	Ebay$_{num}$	**0.73 / 94.1**	12.33 / 100	2.79 / 0	05.98 / 58.8
2	Yandex$_{num}$	**0.89 / 87.7**	15.05 / 96.7	3.30 / 0	15.50 / 02.2
3	reCaptcha$_{phrase}$	**1.02 / 88.0**	20.88 / 91.5	3.03 / 0	N/A

2.6.1 User Study Procedure and Data Collection

We implemented an Android app to experiment with five different challenge response-based liveness detections, where the user either has to read numbers or text presented on the screen, or perform an action in front of the screen. All text-based challenges will have the user read a number of phrases comprised of two to three simple words, and numeric challenges of 6-digit numbers.

It is worth noting that users pronounced all of the numeric or phrase challenges (in plain text or Captcha forms) out loud in our experiments.

To be more specific, our five tested liveness detection based upon the following challenges: 1) two text phrase and one numeric challenges as plaintext; 2) three numeric challenges as Captcha images with reCaptcha, Ebay and Yandex schemes; 3) three text phrase challenges in an animated Captcha images with reCaptcha scheme. In this task, we display challenge words individually by animating (e.g., sliding from left to right) them sequentially with small time delays. The idea behind this approach is to prevent the attacker from extracting the challenge from one single frame, and instead force him/her to extract the Captcha as moving targets. On the other hand, we believe understanding an animated Captcha should be not too much more difficult than solving one at a fixed location for a human being. For this part of our experiment, we used Captcha samples collected by Gao et al. [25] for Ebay and Yandex schemes. To obtain reCaptcha samples that are either purely

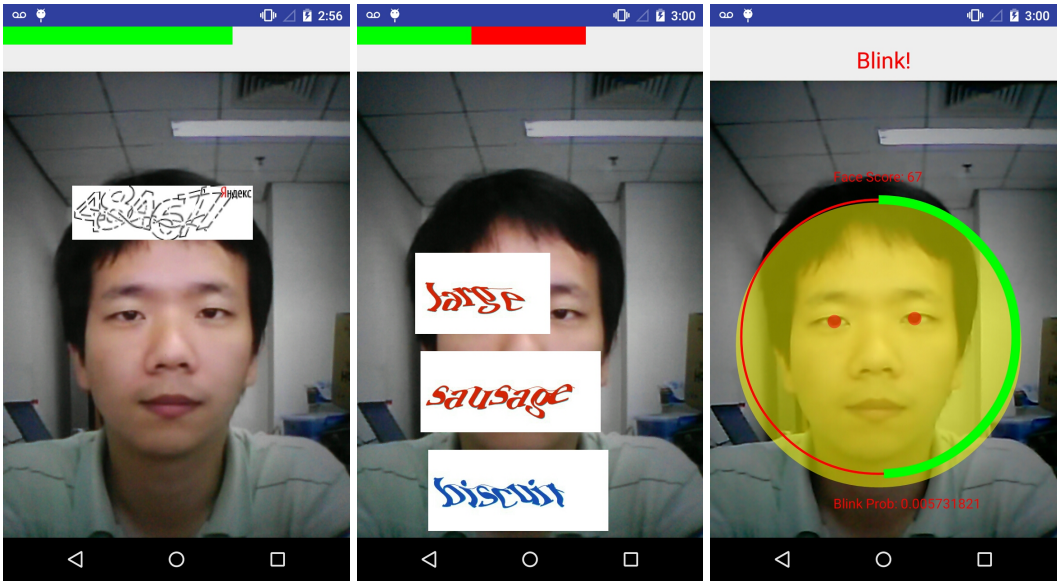


Figure 2.7: Screen shots of our prototype. From left to right; tasks 2, 3 and 4, resp.

numerical or purely text (which are not included in the dataset from [25]), we generated them using *Cool PHP Captcha* tool which creates custom word Captchas in reCaptcha scheme; 4) challenge to blink, and 5) challenge to smile.

To improve the usability of our liveness detection, for challenges 1 to 3, our app will only present one challenge at a time, and we used CMU Pocketsphinx library for real-time speech recognition on mobile devices to know when the user has finished attempting the current challenge (by noticing the stop of utterance), show them whether they're successful before moving on to the next challenge phrase or number. Similarly, for challenges 4 and 5, we used Google's Mobile Vision API to obtain smiling and blinking probability to determine when the user has answered our challenge. Fig. 2.7 shows sample screen shots from our Android app while conducting Captcha challenges and blink detection.

We recruited 31 volunteers for our experiments and had them use our Android app, which has installed on a LG Nexus 4 device we own. At the beginning of our experiment with each participant, we explained the purpose of our experiment and showed them an introductory video about how to use the Android app to complete the tasks. Then we asked each participant to answer 3 rounds of challenges for each of the 5 different kinds of

challenges listed above (i.e., 15 challenges in total). For each challenge, we set a timeout of 10 seconds and considered it a failure and moved on to the next if the participant did not answer the challenge in that time. For the first three types of challenges, we captured the user's audio responses and some facial frames while answering the challenges (like we did in *rt*Captcha), as well as how long it took to answer the challenge and whether the answer was correct. We also compared the facial and voice data from different challenges to determine if it's the face and voice of the same user. For the fourth and fifth challenge types, we only measured and saved blink and smile detection time along with their probability without capturing any video/audio data.

2.6.2 Findings

Before delving into the details of the results from the aforementioned experiment, it is worth noting that participants correctly announced the Captcha challenges with an **89.2%** overall accuracy and **0.93 seconds** overall response time. The accuracy is much higher and the response time is excessively smaller than state-of-the art Captcha breaking algorithms (detailed in further sections). Moreover, 100% of participants' faces and voices are verified correctly with 93.8% (by Face++) and High (by Microsoft) overall confidence values, respectively.

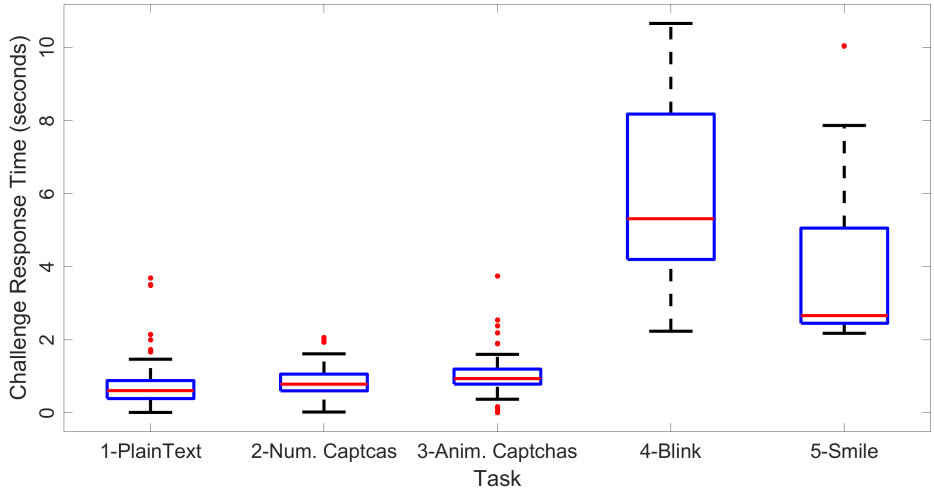


Figure 2.8: Distribution of challenge response times for each tasks.

Fig. 2.8 presents the response time distributions of the participants. While response (and detection) time to any type of challenge that involves the user reading something are below two seconds, the minimum time to give a smile or blink response is higher than the largest measured response time to any of the Captcha challenges (task 2 and 3). The slow detection time for blink and smile may be due to the limitation of our implementation, but we believe they generally require every frame to be analyzed and thus can be more difficult than detecting the utterance of the answer to a text or numerical Captcha challenge. In other words, our experimental results show that Captcha based liveness detection challenges are not going to increase the end-to-end time to authenticate a user over existing smile or blink based challenges.

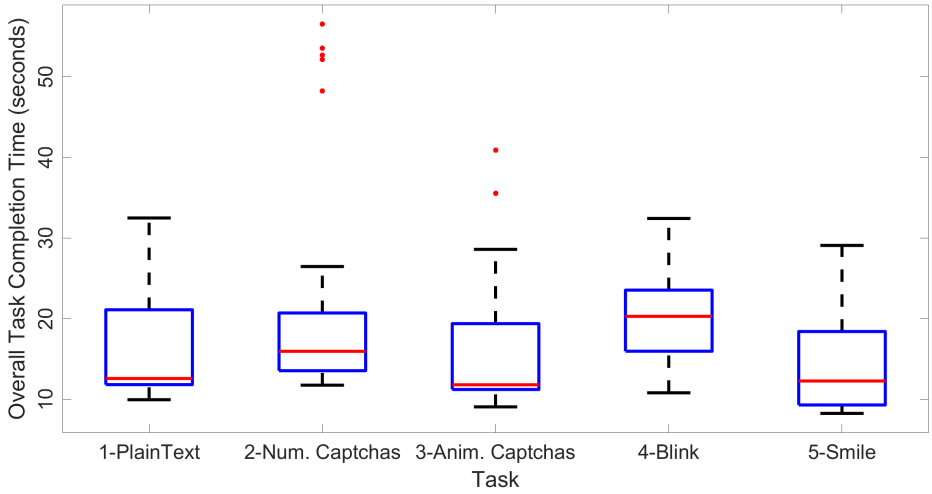


Figure 2.9: Distribution of overall completion times for each tasks.

Finally, Fig. 2.9 shows the overall time to answer all 15 challenges, and we see that there's no significant difference between participants.

In Table 2.3, the left most columns (Human$_{aud}$) give the average response times and recognition accuracies of our participants for each Captcha scheme in challenge type 1 to 3. Also, Fig. 2.10 presents distribution of them for each challenge in tasks 1 to 3. Our results show that participants' response times remain mostly constant over the different types of Captcha schemes tested, and are not affected by the difficulty level of the Captcha. Similarly, recognition accuracies from plain-text and Ebay Captcha challenges to reCaptcha and

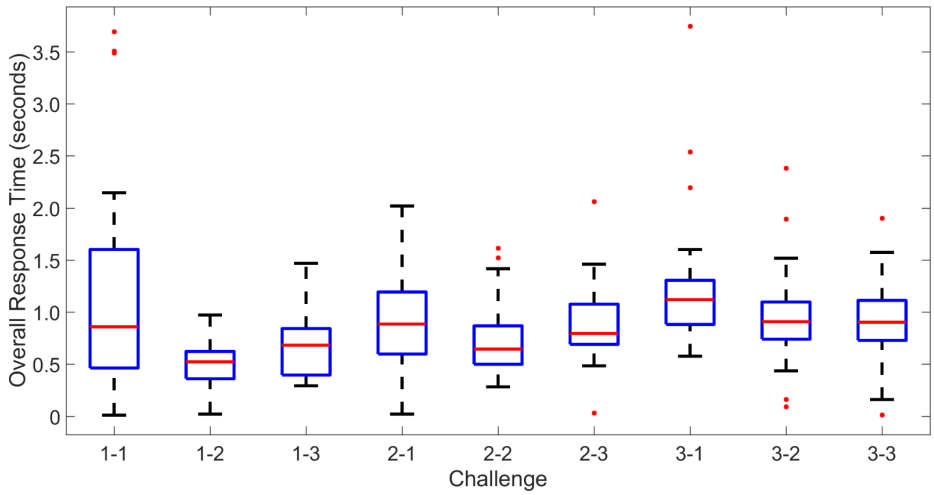


Figure 2.10: Distribution of response times for each challenges in each tasks. (1-1 and 1-2: Plaintext Phrases, 1-3: Plaintext numbers, 2-1: reCaptcha numbers, 2-2: Ebay numbers 2-3: Yandex numbers, 3-1, 3-2 and 3-3: Animated reCaptcha phrases)

Table 2.4: Authentication accuracies by number of trials (%)

Number of trial	Task-1	Task-2	Task-3	Task-4	Task-5
1	90.3	87.1	90.3	80.7	90.3
2/3	100	100	96.8	100	100

Yandex Captchas vary only slightly. Moreover, while numeric Captchas have consistently better accuracies than English phrase-based Captchas, the difference is below 5%.

Fig. 2.10 also shows that there is a slight warm-up effect at the beginning of each tasks; the response times for the first challenge of each task is longer than that for the others. Moreover, when we change the challenge type (e.g. from phrase to numeric at task 1) or the Captcha scheme (e.g. from Ebay to Yandex at task 2), we also observe a slight warm-up effect. On the other hand, since we did not change challenge type or Captcha scheme on task 3, response times decreased in each trial at this task. In any case, even if we have a warm-up effect, the maximum response time to the Captcha challenge is 3.74 seconds, which is still below the execution time of current Captcha-breaking algorithms.

Finally, when a user fails to correctly answer any kind of liveness detection challenge, he/she will be asked to try again. So, in Table 2.4, we measure how many times our participant had to re-try before a successful authentication could be completed under the

different types of challenges. Our results show that in almost all cases, participants needed to try at most two times to successfully respond to any challenge. The only exception happened for one participant under the animated Captcha challenge. When we manually inspected his response, we realized that the problem was caused by the speech recognition algorithm.

2.6.3 Usability Evaluation

We measured the usability and user acceptance rate by asking subjective questions at the end of our user study. Each participant faced two to four questions depending on their answers. We asked the following questions with the stated multiple choices:

1. Have you ever interacted with any kind of facial authentication systems? (Y/N)
2. If yes, what was the challenge it asked? (Blink/Smile/Other)
3. Would you consider using Real Time Captcha in the future? (Y/N)
4. If no, why? a) I don't like Captchas, b) I don't like voice recognition systems, c) I prefer using password protection.

Overall, 87.1% of the participants never have used any kind of facial authentication system and 81.5% of them lean toward an authentication system that offers the proposed liveness detection scheme. The rest do not want to use our framework because of the voice recognition component or Captcha scheme. Finally, 12.9% of the participants have used smile or blink detection-based facial authentication systems, and they stated that they prefer *rt*Captcha. Even though an 84% favorable response from our participants shows promise, we consider it only a preliminary result. As future work, we plan to perform another user study to establish the usability of our system in the general population. This is because: 1) our current user study has a small number of participants, and 2) the participants are from limited diversity of age and background (e.g., mostly university students).

2.7 Security Analysis

In this section, we will first present our analysis to determine how likely it is for an attacker to successfully evade *rt*Captcha and impersonate the user. As mentioned in the threat model, we assume the attacker can compromise the victim phone's kernel, and can have his/her malicious version of the client app used for authenticating with *rt*Captcha. Furthermore, the attacker can also use the victim's phone camera and microphone to collect face and voice samples of the victim, and use available techniques to build accurate models for the victim's face and sound. Thus, when *rt*Captcha presents the attacker with a Captcha, his/her main obstacle in achieving successful authentication is to *solve the Captcha before the authentication session times out*. Once the Captcha is solved, the created facial/voice model of the victim can be used to create video/audio of the victim saying the answer to the Captcha. This fabricated answer can be sent to our authentication server either by injecting it into the system as outputs from the camera and the microphone (through a compromised kernel) or directly into a malicious version of the client app.

Since our system measures the time between when the Captcha is first presented to the time when the user starts to speak, one possible attack against our system is for the attacker to produce an arbitrary "filler" sound (e.g., "errrr") while trying to automatically solve the challenge and then inject the synthesized video. This attack can lead to one of the following scenarios: 1) the "errrr" part is detected by the speech-to-text library, and results in the attacker giving the wrong response, or 2) the "errrr" part is ignored by the speech to text (in this case, we can modify our system to ignore the beginning of an utterance, but instead the beginning of speech recognized by the speech-to-text). Another potential attack against use of the start of speech is for the attacker to focus their effort in identifying/solving the first part of the Captcha. However, we will argue that a main challenge in solving Captcha is to break up the different characters and digits, and thus this attack may not buy the attacker much time.

Table 2.5: Best decoding accuracy and time of generic attacks

Captcha Scheme	Gao et al.[25]		Bursztein et al.[65]	
	Acc. (%)	Time (sec.)	Acc. (%)	Time(sec.)
reCAPTCHA(Old)	7.8	8.06	21.74	7.16
reCAPTCHA	77.2	10.27	19.22	4.59
Yahoo!	5	28.56	3.67	7.95
Baidu	44.2	2.81	54.38	1.9
Wikipedia	23.8	3.74	28.29	N/A
QQ	56	4.95	N/A	N/A
Microsoft	16.2	12.59	N/A	N/A
Amazon	25.8	13.18	N/A	N/A
Taobao	23.4	4.64	N/A	N/A
Sina	9.4	4.83	N/A	N/A
Ebay	58.8	5.98	47.92	2.31
Yandex	2.2	15.5	N/A	N/A

One key to considering the attacker's chance of success is the threshold for session time out; let's call it Th_{legit}. To put it another way, the strength of *rt*Captcha depends on the difference between a Th_{legit} threshold that's long enough for legitimate human users to have good success rate at authentication, versus a Th_{legit} threshold that allows for accurately breaking Captcha using a Captcha-breaking algorithm. Thus, in the following, we will refer back to our experiments in Sect. 2.6.

2.7.1 Setting $Th_{legit} = 5sec$

Participants in our user study responded to 98.57% of the challenges in less than 3 seconds. Furthermore, our results in Sect. 2.6 also show the studied users have an overall accuracy of 87.1% for all tested Captcha schemes, and there seems to be no correlation between their response time and their success rate. In other words, we will not see any significant improvement in the user's rate of successfully answering the Captcha even if we set Th_{legit} significantly higher. Thus, for the rest of our discussion, we'll assume Th_{legit} to be 5 seconds.

2.7.2 Automated Attacks under $Th_{legit} = 5sec$

Now let's consider what is the attacker's chance of breaking our Captcha and successfully generating the video/audio of a victim answering the Captcha with a session time out of 5 seconds. We will base our discussion on different kinds of Captcha breaking methods with different levels of sophistication.

The most primitive Captcha breaking method we have tested is Optical Character Recognition (OCR) based. In particular, we tested the Captcha used in our study against one of the OCR-based Captcha solving websites[11]. As presented in the Attack$_{ocr}$ columns of Table 2.3, the tested site could not solve any of our Captcha challenges. Later on, we investigated if it can successfully decode anything but plain-text lookalike Captcha images without background noise or distortions.

We've also experimented with state-of-the-art Captcha breaking schemes from Gao et al. [25] and Bursztein et al. [65], which are based on character segmentation and Reinforcement learning (RL) respectively. Table 2.5 summarizes their best decoding accuracy and solving times for various schemes on commodity laptops. We consider the work in [25] to be state-of-the-art because it proposes the most generic solution and is the only published work that defeats the Yandex scheme. In Table 2.3 we referred to their system as Attack$_{best}$. While the results in Table 2.5 show that some Captcha schemes can be broken in approximately 3 seconds, the overall recognition accuracies can be very low (while the corresponding accuracies for harder schemes in our user study remain above 85%). Thus, we believe that setting Th_{legit} at 5 seconds gives us a very good safety margin against compromising attacks that could employ even the most advanced Captcha-breaking scheme.

2.7.3 Semi-Automated Attacks

Although in our threat model we stated that any attack that requires human intervention is not going to scale and thus is out of scope, we still will consider the possibility of breaking

[11] http://www.captchatronix.com

Table 2.6: Reported average decoding accuracy and time of typing based human responses to Captcha challenges

Service	Acc. (%)	Time (sec.)	Service	Acc. (%)	Time (sec.)
anti-captcha	99.0	7	2captcha	96.6	10
captchaboss	99.9	8	imagetyperz	99.0	12
deathbycaptcha	95.8	10	9kw.eu	N/A	30

*rt*Captcha using cloud-based, manual Captcha solving services, since this is a commonly used attack method against other Captcha schemes. In particular, attackers may use the authentication device as a proxy and ship Captcha solving tasks to the real human workers. There are many man-powered Captcha-solving services that report high recognition rates, as presented in Table 2.6. We obtained recognition times and accuracies, as advertised on the official website of each service in Table 2.6. While the advertised times are much higher than our 5 seconds Th_{legit} threshold, we also used *2captcha.com* to break the Captcha dataset we used in the user study to obtain real numbers for a fair comparison. The average response times and decoding accuracies of this service under each scheme are presented in Table 2.3 under the Attack$_{typ}$ columns, while the distribution of response times are presented in Fig. 2.11. The average solving time is 19.17 seconds (with 10.75 seconds at minimum) with 96.2% overall solving rate. As such, once again, an attacker trying to launch compromising attacks based on these services will not be able to beat the 5 second threshold, and that is true even if we do not consider other time overheads caused by synthesizer, which has T_{tts}=1.1 seconds (TTS delay time) at best [53], etc.

2.7.4 Other Security Benefits

While the main strength of *rt*Captcha lies in presenting the attacker with a challenge that is difficult to answer automatically, and thus nullifying the advantage they have in being able to generate authentic-looking/sounding video/voice of the victim and inject it into the authentication process at will, *rt*Captcha also comes with a surprising benefit over other liveness detection challenges like blinking and smiling. That is, it is very difficult to cap-

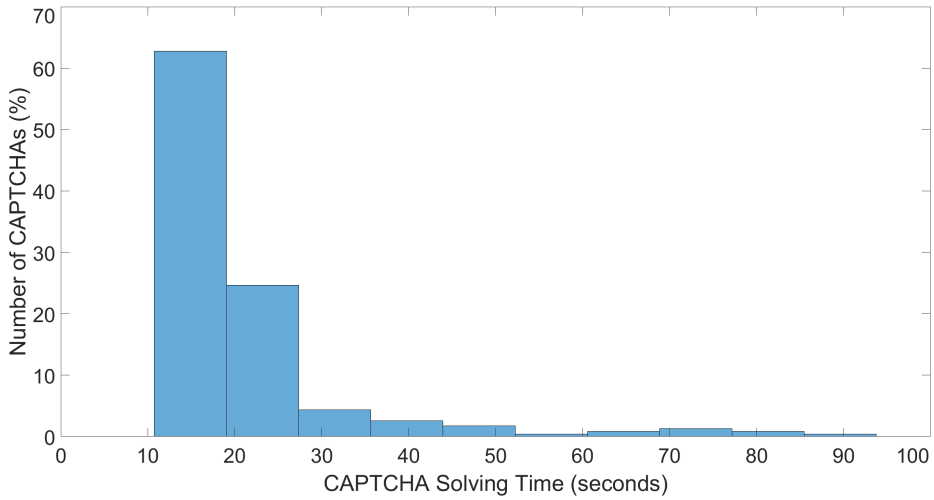


Figure 2.11: Distribution of response times of *2captcha* service for our Captcha database.

ture the user giving out the answer to the right Captcha "just by accident." In particular, liveness challenges that are based on blinking and smiling are very vulnerable to attacks like UI redressing attacks [42], or more advanced attacks like those described in [66]. Under both scenarios, the attacker can drive the legitimate authentication app to a state where it's presenting the user with its liveness detection (either by using Intent, which is harder to control for more than one UI, or using the accessibility service), while covering up the phone's display with an overlay (so the user doesn't know he/she is being attacked). With liveness challenge based on blinking or smiling, this attack is likely to be successful because people naturally blink and smile occasionally, and thus they will provide the answer to the underlying challenge and help the attacker to authenticate them unknowingly. With *rt*Captcha, such an overlay-based attack is unlikely to be successful because it is very unlikely the victim will spell out the answer to the right Captcha by accident while the overlay is obscuring the screen and the underlying app is waiting for a response.

2.8 Discussion and Future Works

Reaching a big diversity in our user study was our limitation as in the previous studies [46]. Since we mainly recruited people around the university, these users were usually more

familiar with the underlying technology with *rt*Captcha. Hence, human performance may vary among other populations, especially among those who rarely use mobile devices or are unfamiliar with Captchas, as stated by Brodic et al. [59].

One of the main security infrastructures in our framework relies on speech recognition since we capture audio response to the Captcha challenges. Hence, the STT algorithm must be robust enough to minimize the false negatives for legitimate user responses. The collected data in our user study involves ambient office, restaurant and outside environments with A/C sound, hums and buzzes, crowd and light traffic sounds. However, our data still have limited background noise variations to test the robustness of used STT method in our experiments. Having said that, we can always use other powerful STT approaches such as Deep Speech 2 by Baidu [61] or cloud-based solutions (instead of CMU Pocketsphinx library) for noisy environments. Moreover, recent advances in lip reading (e.g. LipNet [67]) provide around 95.2% of sentence-level speech recognition accuracy by only using visual content. Combining such an approach with STT would probably give very accurate results to legitimate challenge responses. Moreover, using lip reading-based speech recognition also will increase the usability of the system in a silent environment.

Our future work includes implementing a lip reading method on the Captcha response recognition. Furthermore, future research is required to analyze more data collected from different populations and environments with varying noise types and levels. It also could be required to analyze varying illumination and pose to measure face recognition and verification performance in a real life scenario. However, most of these limitations are related with all audio/visual authentication systems.

Even though we consider presentation attacks out of scope for this work, we believe that by requiring the user to actually say something in order to authenticate, we will create extra challenges for even state-of-the-art presentation attacks. In particular, no matter whether the presentation attack employs static pictures or wearable masks, *the attacker will have difficulty in using those materials to present genuine muscle movement*. This is obviously

true for static pictures, but even for wearable masks, where it is doubtful that one can move lips (without exposing the lips of the person wearing the mask underneath) well enough to appear to be saying the answer to our Captcha challenge. Thus, on top of incorporating lip reading into our system, in the future we also plan to evaluate how this will stop attacks from wearable masks (which should be the state-of-the-art for presentation attacks).

Finally, one can argue that the recently announced Face ID[12] by Apple already provides a robust security mechanism against our threat model, and also, wearable masks. However, our approach still can be applicable through the existing smart phones (estimated around 2.3 billion by the end of 2017[13]) without the requirement of any additional hardware (e.g., depth camera, infrared sensors, etc.).

2.9 Conclusions

Our work outlines several aspects of an audio/visual authentication system and presents a novel and practical approach, called *rt*Captcha to straighten the flaws of existing liveness detection systems. First, our exhaustive analysis on major cloud-based cognitive services (which have a market size of $15 billion[14]) clearly reveals that an applicable and spoof-resistant liveness detection approach is an urgent need. On the other hand, Captcha-based human authentication has been used successfully on the web for more than a decade. Therefore, *rt*Captcha is a suitable fit for this urgent and growing need. Additionally, our user study and comparative threat analysis with its results proves that our scheme constitutes a strong basis against even the most scalable attacks involving the latest audio/visual synthesizers and state-of-the art Captcha-breaking algorithms.

[12] https://www.apple.com/iphone-x/
[13] https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
[14] https://www.biometricupdate.com/biometric-news/biometric-research

CHAPTER 3

PRIVACY PROTECTION IN REMOTE BIOMETRIC AUTHENTICATION

In this chapter, I will present Justitia, a privacy-preserving remote biometric authentication system, that introduces how to use fuzzy biometric inputs as part of cryptographic primitives without loss of accuracy of the underlying off-the-shelf deep learning models. I will then present a novel security assessment technique that considers powerful adversarial models, and could be used as a black-box approach to measure the security of future biometrics-based authentication systems.

3.1 Introduction

Recent advances in deep learning (DL) have finally made the task of face/voice-based biometric verification accurate enough for authentication in terms of *usability and security*: too high false rejection rate (FRR) makes a system unusable, too high false acceptance rate (FAR) makes it insecure. With the core accuracy challenge solved, we now face another problem: *privacy*. Conventional biometric verification requires enrollment data of a client $\mathcal{C}$ to be stored in a remote server $\mathcal{S}$, *unprotected*, for later comparison at authentication time. This data storage raises privacy concerns because if an adversary (or $\mathcal{S}$ itself) is able to gain access, it can cause harm via impersonation or by enabling unwarranted surveillance. Usability of biometric verification is appealing, but collecting biometric information of people in the process is too high of a price to pay [68, 18].

The current solution to privacy, deployed by popular systems like Apple FaceID [69] (or TouchID [70]), is to lock the enrollment template in the client's device under hardware protection (e.g., Trusted Platform Module) [71]. This is cumbersome for protecting online accounts because it permanently binds authentication to the device, meaning the user has to enroll each device separately and cannot authenticate without one.

In this chapter, we study *how to achieve privacy preserving face- and voice-based verification using samples that are cryptographically protected, while maintaining the achieved accuracy of the off-the-shelf DL systems and providing the practicality.* In particular, we aim to provide a way for $\mathcal{C}$ to encrypt and decrypt a secret (registered with $\mathcal{S}$) using its own biometric material such that $\mathcal{C}$ can authenticate to $\mathcal{S}$ by proving he has the secret, *but the biometric data of $\mathcal{C}$ is never leaked to $\mathcal{S}$ in the processes of enrollment and authentication*. If we are successful in achieving our goal, we can eliminate the privacy risk of face- and voice-based authentication.

In a typical password-based authentication, $\mathcal{S}$ hashes the registered passwords to protect them against data-breaches. However, due to the presence of some noise (e.g., different lighting, ambient sounds, imprecise scans, etc.), $\mathcal{S}$ cannot hash the biometrics, then compare with the hashes of those captured at the authentication time for an exact match. In response, researchers propose different forms of *fuzzy cryptography*, also used in this work, to protect biometrics in a manner similar to hashing [72].

In this context, Dodis et al. [73] introduce the notion of fuzzy extractors (FE), which is the first and broad formal cryptographic primitive in its kind, but earlier works produced some of the first constructions (e.g., fuzzy commitment and fuzzy vault) [74, 75]. The main idea of FE is to employ error correction schemes to recover an enrollment-established *secret* from non-sensitive helper data using $\mathcal{C}$'s biometric sample obtained at authentication time. Hence, if $\mathcal{C}$ can recover the secret successfully, he can use it to prove his identity to $\mathcal{S}$, without revealing his biometric data to $\mathcal{S}$ (see Sect. 3.3). Note that, the secret is encrypted by $\mathcal{C}$ with multiple keys derived from his biometric data, and all encryptions are kept as part of the helper data. That is how FE i) protects privacy by having $\mathcal{S}$ handles only the helper data, and ii) provides security for our scheme by guaranteeing that only the biometric data "close enough" to the one used to encrypt the secret can decrypt it (see Sect. 3.4).

However, existing bio-template protection schemes, including FE, have two major drawbacks in practice for our focus. First, as many of them employ i) hand-crafted fea-

ture extraction methods for a specific biometric modality (e.g., fingerprint or iris) or ii) custom-built distance metrics, they are not applicable to the DL-generated embeddings of face and voice biometrics. Second, as [76] points out, despite almost 20 years of research, many of them (e.g., winners in [77] and [78]) could not meet the accuracy requirements ($FAR \leq 0.002\%$ and $FRR \leq 10\%$) of industrial systems for practical use [79]. Furthermore, they achieve low security guarantees against impersonation attacks. Though the protocol from [72] improves the security of FE-based systems (relative to the database size), it requires $\mathcal{C}$ to send his bio-template to $\mathcal{S}$ and put strong trust assumptions on $\mathcal{S}$, which are also against our focus.

3.1.1 Our Contributions

To our knowledge, there is no privacy-preserving scheme, for DL-based face and voice biometrics, meeting the requirements of such a remote authentication system in the focus of this work. In this paper, we build a novel pipeline, Justitia, on top of *reusable fuzzy-extractors* (SSF-FE[1]) from [80], and state-of-the art DL systems for face and voice biometrics.

Challenges & our solutions

We solve two major challenges: "*space mapping*" to solve compatibility problem between DL and SSF-FE, and "*noise handling*" to solve the SSF-FE's efficiency problem.

Space mapping The fundamental problem in providing privacy protection on top of DL-based biometric matching is that of a "round hole, square peg" situation. On the one hand, DL models work in grid-like spaces (e.g., Euclidean or cosine) in order to leverage stationarity and compositionality like properties of the data through local statistics [81]. On the other hand, SSF-FE, for protecting biometric data, works in Hamming space via bit-wise operations.

[1]Named after the last names of the author of [80], and used interchangeably with FE.

To this end, one of the core technical contributions of this work is bridging these spaces while still attesting to both the security of the cryptography, protecting the biometric data, and the accuracy of the underlying DL system.

Noise handling SSF-FE can handle the noise in biometrics, but the size of the required helper data of FE (to guarantee secret recovery) increases exponentially with the amount of noise we want to tolerate. As a result, each of eliminated noisy bit, from the biometric representations in Hamming space, will exponentially decrease i) the data transmitted between $\mathcal{C}$ and $\mathcal{S}$ (and storage space in $\mathcal{S}$), and consequently, ii) the response time of SSF-FE.

Hence, the resolution of above "round hole, square peg" problem also helps in applying further noise reduction (NR) techniques in Hamming space (see Sect. 3.2.4). Additionally, the client app can enforce $\mathcal{C}$ to scan only high-fidelity biometric samples, which cause less amount of noisy bits, by applying input quality filters (QF) at enrollment and authentication times (see Sect. 3.2.1). Note that, this also maintains the accuracy of underlying DL system

Remote authentication protocol

Now, $\mathcal{C}$ can prove that he has the secret, as followed. $\mathcal{C}$ sends the hash of the secret (e.g., computed via a pseudo-random function) to $\mathcal{S}$ at enrollment time. Similarly, $\mathcal{C}$ sends the hash of the recovered secret to $\mathcal{S}$ at authentication time. Then, $\mathcal{S}$ compares both hashes to verify if $\mathcal{C}$ has the secret,without learning nothing about the biometrics of $\mathcal{C}$. Note that, this also makes Justitia compatible with the password-based schemes as $\mathcal{S}$ handles only the hashes of the secret.

Improving security

We show how to improve the security of our design, by i) combining face and voice biometrics in a way that does not suffer from the accuracy, and ii) using Justitia as a second factor as it is compatible with the password-based authentication. For instance, $\mathcal{C}$ can send

the hash of the concatenation of his secret and password to $\mathcal{S}$, to improve the security for sensitive authentications. Note that, this also brings additive protection against targeted biometric attacks. Moreover, in our design, combining face and voice could be handled seamlessly, by having the client app captures a short video of $\mathcal{C}$ while he is pronouncing a short phrase (e.g., 2-3 seconds) in front of the camera.

Improving usability

Scanning biometrics (especially in a noisy environment) for every single authentication request presents usability problems. We can prevent this by not using the biometric-based authentication for every authentication attempts. Instead, we can tie the client's identity to the mobile device used for biometric-based authentication. From then on, day-to-day authentications for non-sensitive operations can be performed using standard one-click device-based authentication, i.e., in Fast IDentity Online (FIDO) approach [82].

Implementation

We implement an end-to-end prototype from scratch, to measure the accuracy, performance and usability of our design for face- and combined-biometrics-based remote authentications. We use standard cryptographic primitives and state-of-the-art DL models (i.e., FaceNet [24] and DeepSpeaker [83]) in our implementation. We also implement above one-click authentication protocol to simulate the day-to-day usage in our user study.

Evaluation results

To measure accuracy of our end-to-end design, we compare directly against error rates of a non-private (plaintext) pipeline. Justitia achieves the same low error rates (0.33% FRR, 0% FAR) as the plaintext system does on a subset of the YouTube Faces (YTF) benchmark. Note that, we also evaluate our system's FAR under an impersonation attack scenario, by assuming a powerful adversary, as 0% FAR might be the result of not having

enough negative samples in YTF dataset. Moreover, through the proposed face&voice-based pipeline, Justitia achieves (1.32% FRR, 0% FAR) compared to (1.14% FRR, 0% FAR) errors of plaintext system. Finally, Justitia achieves similar response times, on new device enrollment/renewal, compared to the prior plaintext systems.

Threat model & security analysis

We evaluate the security of our design in the semi-honest $\mathcal{S}$ model, where $\mathcal{S}$ follows the pipeline while trying to obtain an encrypted secret of an enrolled person in the database. Hence, we consider only large-scale attacks, as such the adversary compromising the database has the same advantage with $\mathcal{S}$. Note that, no greedy/gradient descent search is possible as the data is kept under encryption in $\mathcal{S}$.

We first discuss the privacy and security guarantees, achieved by underlying standard crypto primitives, of our design (see Sect. 3.4). Then, we support them through systematic security evaluations by i) using existing approaches and ii) proposing a novel method considering more powerful attacks than existing work.

Previous work either conducts empirical attack simulations [84] or statistical estimation techniques [85, 86]. While the latter may fall short in modeling a broader attack space due to the limited data, the former assumes a weak adversary, which does not narrow down the attack space as it only generates random inputs. They measure from 25 (by [86]) to (more than) 40 bits of security on Justitia.

In this work, we consider an adversary that can leverage from the domain knowledge, while probing random inputs (as impostors) to our pipeline. In particular, we measure the probability of impersonating one of the enrolled users from YTF dataset while brute-forcing through (impostor) faces from photorealistically synthesized subjects. Empirically, we only observe *2* falsely accepted faces out of 60 *million* brute force attempts, which matches with the statistically-backed estimate from [86]: 1 impersonation per 24.6 million ($\tilde{}2^{25}$) attempts. Furthermore, combining face and voice boosts the security of Justitia to

˜33 bits.

3.1.2 Related Work

Other than fuzzy cryptography, biometric authentication (along with the related and alternative approaches [87, 88, 89, 90, 91, 92]) has long been studied for different application scenarios and modalities, by many works and industrial systems. These are based on multi-party computation, Homomorphic encryption and hardware tokens, which do not accommodate the focus of this work, as they require $\mathcal{S}$ holding plaintext database, $\mathcal{C}$ handling key-management or possessing the token all the time.

Multi-party computation

There are techniques protecting queries against a biometric database in a secure multi-party computation (MPC) environment [93, 94, 95, 96, 97]. However, they solve a fundamentally different and easier problem, one where the server has access to the plaintext face.

Homomorphic encryption

HE allows computing arbitrary functions, including biometric authentication, under encryption [98, 99]. Though advances in HE performance [100] are encouraging, the fundamental issue is key management and trust [101]. Indeed, while it is easy to compute *under encryption* the `yes/no` output of authentication decrypting the result requires the decryption key. We want to avoid client-side storage of the keys. Furthermore, storing the key at the server defeats the purpose of the HE, as it decrypts all data (e.g., input biometrics). Protocols from [102, 103] solve this problem by using two servers to isolate the biometric data and the key. Note that, this introduces additional strong trust assumptions, which we avoid in our work.

Token-based authentication

Schemes from [104, 105, 106] employ a second factor, e.g., a hardware token, to apply a reproducible distortion of the biometric data. The distorted data is stored by the server, and the user maintains the secret. At authentication time, the user generates another distorted biometric reading using the secret, which will be matched against the stored data. Note that the second factor must not be entrusted to the server (hence must be maintained by the user), since some of the distortion is reversible. Loss of this secret token presents a recovery problem.

3.1.3 Out-of-Scope Assumptions

We require the client app having access a single *global* DL model that is not tailored to any particular $\mathcal{C}$. Hence, we do not consider such membership inference [107] or model inversion [108] attacks to be a privacy threat in the context of securing *authentication*.

We do not consider any targeted impersonation attack, either by directly feeding stolen biometric into the client app, or by any perturbation on the input of the DL model. Potential defenses to such attacks include better liveness detection (LD) [16, 109], strengthening the DL model against perturbations [110, 111], and coupling with passwords as mentioned above, which are also not in our focus.

3.1.4 Summary of Our Contributions

1. We propose a novel pipeline, Justitia, accommodating privacy-protection on DL-generated embeddings of face/voice biometrics, by solving the space mapping problem between underlying deep learning and cryptographic methods.

2. We introduce efficient noise handling techniques to maintain the accuracy of the respective plaintext systems.

3. We propose methods to combine multiple biometrics without suffering from accuracy,

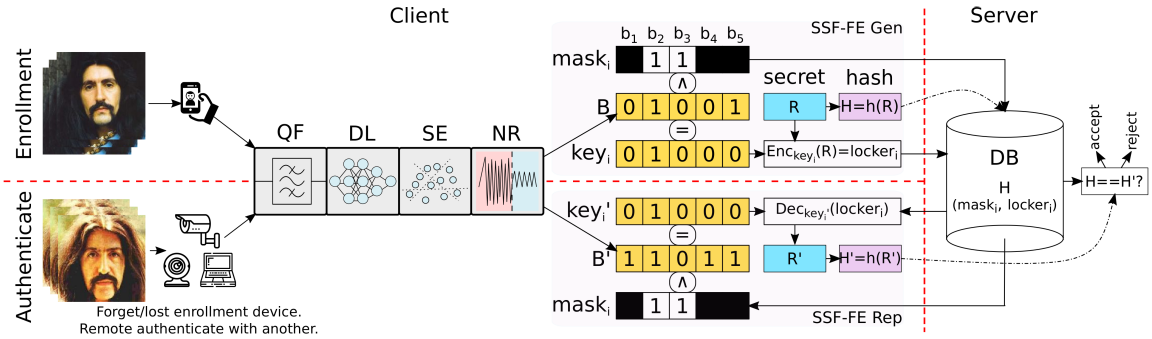


Figure 3.1: A remote authentication protocol based on Justitia. Gen and Rep algorithms of SSF-FE are shown for one mask and bio-bit vectors (B, B'), with five bits $(b_1, \ldots, b_5)$, from enrollment and authentication, resp. Hamming distance $dist(B, B')$ is 2 bits. The real implementation uses multiple masks and longer bit vectors to produce multiple secure lockers. Only the masks and lockers are stored on $\mathcal{S}$. Hash H is equal to H' as the secrets R and R' are the same in this example.

and accompany to passwords as a second factor to increase the security.

4. We improve usability by accommodating standard one-click device-based authentication for non-sensitive operations.

5. We systematically evaluate our security through four different approaches from prior art. We also propose a novel black-box security assessment method, which could be used for measuring the security of future face-based authentication systems.

6. We build and evaluate a prototype on standard benchmarks.

7. We conduct a user study to measure usability of our prototype for sensitive authentication scenarios and day-to-day usage.

8. We achieve comparable user acceptance to known two-factor authentication (2FA) systems.

3.2 Building Blocks

In this section, we discuss the ideas behind Justitia pipeline, which is depicted in Fig. 3.1. As mentioned before, we consider liveness detection out-of-scope for this work. Hence, we

assume all inputs are validated through an off-the-shelf, privacy preserving, LD process. Our protocols are instantiated upon Justitia in Sect. 3.3.

3.2.1 Input Quality Filtering

Compared to other modalities like fingerprints or iris, the use of face/voice for authentication presents unique opportunities and challenges. On the one hand, since almost all mobile devices and laptops come with both microphone and camera, capturing face/voice samples are much more readily deployable. On the other hand, the accuracy can vary significantly based on how these samples are captured in the first place. For example, it should be obvious that voice based authentication will not work well in a noisy environment. Similarly, face based authentication will not work well with faces captured in i) high (or low) lit environment or background, ii) low resolution, high (or low) contrast or brightness, iii) occluded or blurry conditions, and iv) with too much pitch, yaw or roll angles.

Overall, to guarantee the bio-bit vectors have high signal-noise ratio, $\mathcal{C}$ locally checks the quality of the input biometric samples, and provide the user feedback on what to do to improve the sample quality if the input is deemed too poor quality, while capturing samples with the client app. We design the following filters to utilize sensors available on most mobile devices and avoid relying on advanced sensors (e.g., infrared camera, depth sensor etc.). In particular, the client app will:

1. detect the face orientation and retain those within certain facial angle thresholds through pitch, yaw and roll axis,

2. make a color space conversion from RGB to YUV, where Y refers to brightness and UV refers to color components, then retain faces within a certain brightness thresholds,

3. compute a statistical quality assessment score over a pre-trained model representing known distortions [112], then retain faces having a score within certain thresholds.

Note that, applying our pipeline is requested rarely in improved-usability scenario (see

Sect. 3.3.3). Besides, similar to existing authentication systems, we may allow $\mathcal{C}$ to authenticate, e.g., once a week, as long as they are accessing the controlled resource using the same device. Thus, the chance that $\mathcal{C}$ will have to conduct QF check at a noisy environment is significantly reduced. Moreover, in Sect. 3.7.1, we also make a preliminary analysis to show the impact of QF checks on the usability, by measuring the average time of our participants capturing qualifying facial samples.

Note that, face and voice quality assessment have been studied for different application scenarios [113, 114, 115, 116, 117]. Note that, we can leverage from these methods to improve the Step 3. However, our technique avoids non-qualifying samples even before requiring to apply these models. In Sect. 3.6.3, we show that the proposed approach can significantly reduce the error rates (e.g., FRR/FAR decrease from 2.01/2.02% to 0.33/0%).

3.2.2 Deep Learning (DL)

In this step, $\mathcal{C}$ locally converts raw biometric readings, retained in QF layer, into d-dimensional feature vector representations (e.g., *embedding vectors*), by using the state-of-the-art DL systems (e.g., FaceNet [24] and DeepSpeaker [83]), with the guarantee that *two such vectors will only be close in Euclidean distance (or cosine similarity) iff they are of the same person*.

3.2.3 Signal Extraction (SE)

Though the Euclidean space of DL accurately captures the statistical properties of the raw input data, unfortunately, even the two consequent biometric scans of a person will not result the same embeddings due to the continuous nature of the Euclidean space. Hence, in order to accommodate SSF-FE, $\mathcal{C}$ locally translates d-dimensional embeddings into $\mathcal{L}$-bit binary codes, by using Super-Bit Locality Sensitive Hash (SBLSH) [118].

SBLSH is built on top of Sign-Random-Projection LSH (SRP-LSH) [119] , which turns input vectors into *one-bit hash* such that if two input vectors (from enrollment and authen-

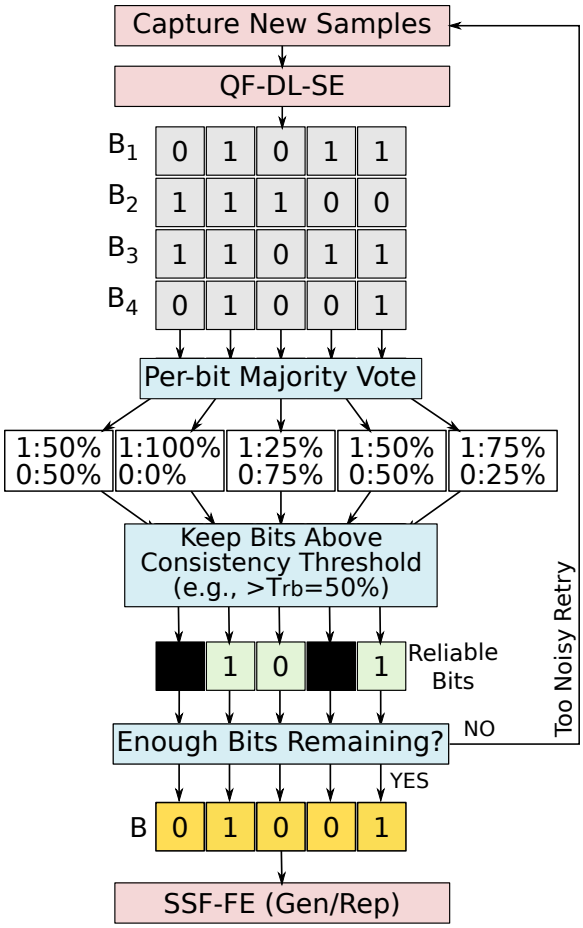


Figure 3.2: NR. Majority voting keeps bits, that are above a predetermined threshold τ_{rb}. B_i is the i^{th} bio. scan's bit vector.

tication phases) are close in angular distance, it is likely that their SRP-LSH will be the same. In particular, for given embedding vector x and a uniformly-sampled vector v (both in d-dimension), SRP-LSH is defined as $h_v(x) = sgn(v^T x)$, where $sgn(.)$ is a sign function (i.e. 1 if the input is greater than or equal to 0, otherwise 0). Then, to turn x into $\mathcal{L}$-bit binary codes, SBLSH independently samples $\{v_1, ..., v_\mathcal{L}\}$ vectors, and for each $i \in [\mathcal{L}]$, calls $h_{v_i}(x)$. Note that, these vectors are public to every clients.

3.2.4 Noise Reduction (NR)

Inspired by [80, 120], $\mathcal{C}$ can collect multiple biometric samples at enrollment and authentication attempts in order to perform noise removal. This can be done very seamlessly for

face/voice samples because for face, $\mathcal{C}$ can record a very short video of his face in front of the camera. Since people normally will not be able to hold completely still while recording this video, we can treat each individual frame in the video as a different sample. Similarly, $\mathcal{C}$ can collect sufficient voice samples (utterances) from a short audio recording (e.g., 2-3 second-long sentence [121, 122]).

In this step, $\mathcal{C}$ takes $\mathcal{L}$-bit binary codes, generated in the SE step through multiple (e.g., $\mathcal{N}_{br}$ =10) biometric readings, and majority vote over each bit. If a certain amount of them agree (e.g., at least τ_{rb} =80 percent), $\mathcal{C}$ keeps the bit. Otherwise, it is canceled. If too many bits are canceled (e.g., more than half of the bits), we assume that the captured set of samples is too noisy, and $\mathcal{C}$ will restart from the sample capturing process. After eliminating noisy bits through multiple samples, the NR layer gives the residual, representative, bio-bit vector B to Gen algorithm of SSF-FE in enrollment phase. Similarly, B' is generated and given to Rep algorithm of SSF-FE in authentication phase. The steps are visualized in Fig. 3.2.

3.2.5 Fuzzy Extractor from Random Oracle

A (m, ℓ, t, ϵ)-FE scheme is defined for input space $\mathcal{M}$, distance function *dist*, and a pair of algorithms (Gen, Rep) ("generate" and "reproduce"). Gen, on input $B \in \mathcal{M}$ from the input space with entropy m outputs an extracted string $R \in \{0,1\}^{\ell}$ and a helper string $P \in \{0,1\}^{*}$. The FE correctness property ensures reconstruction of R from close biometric such that if $dist(B, B') \leq t$ and $(R, P) \leftarrow Gen(B)$, then $Rep(B', P) = R$. Finally, the crucial privacy property guarantees that the string R is close (expressed in terms of ϵ) to uniform, even if the helper data P is public.

In our design, we use a specific FE under Hamming space extension, the *reusable FE* (SSF-FE) from [80]. Since *it is based on the Random Oracle (RO) model, its security is based on the well accepted/understood hardness of reversing hash/encryption functions*. In contrast, other systems only guarantee the mathematical complexity of breaking their

security and related works have demonstrated some known cases of feasible attacks [123, 124, 125].

Our pipeline uses digital lockers [126] while constructing SSF-FE. Fig. 3.1 illustrates at a high level how a digital locker works. The goal is to allow input bio-bit vectors (B, B'), which are close in Hamming distance (e.g., 2 bits in the example), to be used to generate the same keys $key_i = key'_i$ for encrypting/decrypting a *secret R*. The encrypted secret *R* is called "locker" as it can be "locked" and "unlocked" by the two bit vectors. It achieves this goal (generating the same key) by *masking (i.e. turning to a constant zero)* the error bits (e.g., (b_1, b_4) in the example). An astute reader will note that even if we can guarantee the Hamming distance between B and B' to be at most 2, the mask in the example may not always be able to eliminate all errors. Thus, multiple sets of $(mask_i, locker_i)$, where $i \in [N]$, are needed to achieve high probability (i.e., $1 - \epsilon = 1 - 10^{-4}$ in our construction) in regenerating the right keys and unlocking the secret R from at least one of the lockers. Note that, *N increases exponentially with t*, where N is the number of $(mask, locker)$ pairs needed to guarantee successfully unlocking the R, and t is the amount of noise we want to tolerate (i.e. the maximum Hamming distance between B and B', used for locking and unlocking R, resp.)

3.3 Instantiating protocols

In this section, we explain how we construct our enrollment and authentication protocols, which are built on top of Justitia pipeline, for a remote authentication scenario. We also discuss improving the i) security and ii) usability of such a remote authentication system, by incorporating our design with i) passwords in a 2FA scheme and ii) standard challenge-response schemes, respectively.

3.3.1 Enrollment and Authentication

Fig. 3.3 and Fig. 3.4, present our constructions of enrollment and authentication protocols, respectively. Note that, the sets of biometric readings (BR, BR'), captured by the client app, are from the domain $\mathcal{D}$. We set the optimal parameters of our design in Sect.3.6.2.

Constructing digital lockers

We construct digital lockers on top of Hash-based Message Authentication Codes (`HMAC`), namely `HMAC-SHA256`, such that the encryption function in the Gen method of SSF-FE (e.g., Enc in Fig. 3.1) is implemented as $locker_i = \texttt{HMAC}(nonce_i, key_i) \oplus (0^\lambda \mid\mid R)$, and decryption in the Rep (e.g., Dec in Fig. 3.1) is implemented as $0^\lambda \mid\mid R = \texttt{HMAC}(nonce_i, key'_i) \oplus locker_i$.

Recovering the secret on the client-side

As noticed above, $\mathcal{C}$ concatenates 0^λ, where λ is a security parameter, before encrypting the secret R into a locker at the enrollment time so that it can verify a correct decryption of $R' = R$ at the authentication time.

Authenticating $\mathcal{C}$ on the server-side

As presented in Fig. 3.3 and Fig. 3.4, in our pipeline, only $\mathcal{C}$ handles the secret R (and recovered R') as it executes Gen and Rep algorithms of SSF-FE, respectively. On the other hand, $\mathcal{S}$ handles only the helper data $P_i = (mask_i, nonce_i, locker_i)$, where $i \in [N]$, and the hashes (H and H') of the secrets. That is, given a publicly known pseudo-random hash function h, $\mathcal{C}$ computes and sends $H = h(R)$ to $\mathcal{S}$ at enrollment time. Then, at authentication time, if $\mathcal{C}$ sends $H' = h(R')$ s.t. $H' = H$, $\mathcal{S}$ remotely authenticates $\mathcal{C}$, without seeing $\mathcal{C}$'s biometrics.

Inputs: $\mathcal{C}$ captures a set of biometric readings $BR = \{BR_1, \ldots, BR_{\mathbb{N}_{br}}\}$, where $BR_i \in \mathcal{D}$, by using the client app.

1. **[Quality Filter]** $\mathcal{C}$ retains only the biometric readings from BR passing through a set of quality filters.
2. **[DL-SE-NR]** $\mathcal{C}$ computes $B = f(BR)$, using publicly defined Justitia functions $f = NR \circ SE \circ DL : \mathcal{D} \rightarrow \{0,1\}^{\mathcal{L}}$.
3. **[SSF-FE Gen]** $\mathcal{C}$ samples a random secret ***R***, and $\forall i \in [N]$:
   (a) samples a random $mask_i = \{0,1\}^{\mathcal{L}}$, which includes ℓ 1-bit,
   (b) computes $key_i = mask_i \wedge B$,
   (c) samples a random $nonce_i = \{0,1\}^{\mathcal{L}}$, and computes $locker_i = HMAC(nonce_i, key_i) \oplus (0^{\lambda} \,||\, R)$, λ is a security parameter.
4. $\mathcal{C}$ computes the hash $H = h(R)$, where h is a publicly defined pseudo-random hash function.
5. $\mathcal{C}$ sends set $\{P_i = (mask_i, nonce_i, locker_i)\}_{i \in [N]}$, and H to $\mathcal{S}$.

Figure 3.3: Enrollment Protocol using Justitia.

3.3.2 Improving Security

Using face & voice together

Building the identity of an individual by combining his/her multiple biometric traits is a common practice to increase the security of authentication systems [127, 128, 129, 130, 131, 132, 133]. In our case, we achieve this combination of face and voice seamlessly by having the client app capture a short video of $\mathcal{C}$ while he is pronouncing short phrases in front of the camera. We combine these modalities as in the following conjunction rule [129]. If $\mathcal{S}$ verifies the hash of secret, recovered by the $\mathcal{C}$'s face, then it tries to verify the one recovered by $\mathcal{C}$'s voice. Since, this rule, on the other hand, naturally increases the FRR (relative to errors in both modalities), we take extra steps to handle this while combining them. In particular, the client app allows $\mathcal{C}$ to authenticate himself through multiple attempts (e.g., up to 3 times) if he is rejected, while he prefers using both face and voice in

Inputs: $\mathcal{S}$ sends the helper data $\{P_i = (mask_i, nonce_i, locker_i)\}_{i\in[N]}$ to $\mathcal{C}$. $\mathcal{C}$ captures $BR' = \{BR'_1, \ldots, BR'_{\mathcal{N}_{br}}\}$, where $BR'_i \in \mathcal{D}$.

1. **[Quality Filter]** Repeat the Step 1 of Fig. 3.3 for BR'.
2. **[DL-SE-NR]** $\mathcal{C}$ computes $B' = f(BR')$ as in Step 2 of Fig. 3.3.
3. **[SSF-FE Rep]** For each $i \in [N]$, $\mathcal{C}$:
   (a) computes $key'_i = mask_i \wedge B'$,
   (b) computes $p_i = HMAC(nonce_i, key'_i) \oplus locker_i$.
   (c) if $p_i[1:\lambda] = 0^\lambda$, then return $R = p_i[\lambda+1 : \lambda+\ell]$.
4. $\mathcal{C}$ obtains $R' = R$ if Step 3c return R, otherwise it gets $R' = \perp$.
5. $\mathcal{C}$ computes the hash $H' = h(R')$, and sends H' to $\mathcal{S}$.
6. **[$\mathcal{S}$ auth.]** If $H = H'$, $\mathcal{S}$ authenticates $\mathcal{C}$, otherwise it rejects $\mathcal{C}$.

Figure 3.4: Remote Authentication Protocol using Justitia.

our design.

Strengthening password-based authentication

For sensitive authentications, clients may prefer using multiple factors, like biometrics and passwords in a typical 2FA scheme. This improves security in two ways. That is, passwords protect users against targeted biometric attacks, and biometrics protect passwords against typical attacks applied to a stolen password database [134, 135].

In our design, for instance, $\mathcal{C}$ can first hash his secret after concatenating with a password (e.g., $H = h(R \,||\, password)$), and send it to $\mathcal{S}$ at both enrollment and authentication times. That is how we increase the security guarantee of our design relatively with the chosen password. To deploy this, we should only modify the client app without touching the server-side. Note that, passwords could be included into protocols from Fig. 3.3 and Fig. 3.4 in different ways. However, we will not introduce a detailed 2FA protocol in this work.

3.3.3 Improving Usability

Scanning biometrics (especially in a noisy environment) for every single authentication request presents usability problems. We can avoid a lot of these issues by *not using the biometric-based authentication in Fig. 3.4 for every authentication attempts.* Instead, at the enrollment time, we can tie the user's identity to the mobile device used for the enrollment. From then on, day-to-day authentications for non-sensitive operations can be performed using standard one-click device-based authentication (i.e. the person with the registered device can authenticated as the user). In this set up, we will only need to perform the biometric-based authentication in Fig. 3.4 when the user need to register a new device (i.e. he will authenticate using his biometric on the new device, and requesting to register this new device; if the authentication is successful, his identity will be tied to the new device). We implement this usability improvement by using OpenID connect [136] (i.e. the FIDO approach), and evaluate its user acceptance in Sect. 3.7.2.

3.4 Security Discussion of Justitia

In this section, we present a high level argument for the security of our design, which comprise of the enrollment and authentication protocols presented in Fig. 3.3 and Fig. 3.4, under the assumption that $\mathcal{S}$ is honest but curious, i.e. $\mathcal{S}$ follows the steps of the enrollment and recovery pipelines, but tries to learn additional information about $\mathcal{C}$ through the stored data in its database. *This setting is also applicable in modelling an attacker who compromised the database, since he will have access to what's available to $\mathcal{S}$.*

In the following, we will present a high level explanation of how underlying crypto primitives, including SSF-FE and hashes from PRF, protect the privacy of $\mathcal{C}$'s biometric data, as well as how they guarantee the security of our authentication scheme. Recall that, no greedy/gradient descent search is possible since $\mathcal{S}$ only holds the encryption or hash of the secret, thus attackers can only brute force. In Sect. 3.9, we present the achieved security

in practice through diverse analysis. We refer interested readers to [80, 126] for the formal security proofs of the security and privacy of SSF-FE.

3.4.1 Privacy

The privacy goal of the proposed protocols is to protect the input bio-bit vectors B from the adversary. As noticed in the protocols, for each $\mathcal{C}$, $\mathcal{S}$ only holds the helper data for SSF-FE (i.e., $P_i = (mask_i, nonce_i, locker_i)$) and H, the hash of the secret. As such, it is safe to make both of them public in terms of privacy, because: i) H has nothing related with B, ii) the masks are entirely random and are not derived from any bit from B, iii) the lockers are encryptions of the secret with different keys, which are the masked B. The encryption algorithm guarantees that nothing about either the keys or the locked secret will be revealed unless the adversary guesses the right key, and each key is $\ell = 128$ bits.

3.4.2 Security

Our security goal is to prevent the adversary (who does not have the correct input bio-bit vector) from extracting the secret of any $\mathcal{C}$. To do this, he has to i) reverse the hash $H = h(R)$, whose security relies on reversing the hash function h from a PRF family, or ii) decrypt at least a SSF-FE locker. While the encryption algorithm should provide standard defense against any leakage through the ciphertext, the protection of a locker is only as strong as the key. In particular, even if we have sufficient unmasked bits in the key, the ciphertext thus generated is still vulnerable to brute force attacks if the input bio-bit vector is very low in entropy. In Sect. 3.9, we provide our analysis of how much security we can achieve in practice.

3.4.3 Unlinkability (Reusability)

According to ISO/IEC-24745 standard [137], one of the requirements of biometric authentication systems is unlinkability of biometric templates across different databases [138].

Our design meets this through the reusability of SSF-FE, as such it is guaranteed in the same manner with the privacy of SSF-FE. In particular, since it is the $\mathcal{C}$ who chooses which secrets and masks to use across different servers. Additionally, none of the $(mask, nonce, locker)$ pairs from the servers, leak any information about the raw biometric data since each locker is the encryption of a secret under a hash function modeled as RO. Hence, colluding servers cannot combine a pair of helper data or hashed secrets of $\mathcal{C}$ to obtain extra information about the $\mathcal{C}$'s raw biometric data. We refer the reader to [139, 140] for a detailed analysis of why reusability is hard to achieve in existing secure sketches/FE schemes that are *not based on the RO assumption*.

3.5 Implementation Details

3.5.1 Prototype Implementation

We implement a prototype system where $\mathcal{C}$ is an Android app and communicates with $\mathcal{S}$ through the REST API. For the DL layer, we use a pre-trained model of FaceNet [2] to extract facial-embeddings. This model is pre-trained on the MSCeleb1M dataset, including 8 million faces from around 100 thousand identities [141]. To infer voice embeddings, we train our own model using an open-source implementation [3] of DeepSpeaker, as introduced in [83]. We use VoxCeleb dataset, including 100 thousand of English utterances from 1,251 identities, to train our model [122]. For reproducibility, we open-source our implementation[4], and clearly present and explain all the parameters in Sect. 3.6.2.

3.5.2 Environment

For evaluating our prototype, $\mathcal{C}$ runs on four different Android mobile phones (LG Nexus 5, Xiaomi Redmi Note 5 Pro, One Plus 6 and Huawei Mate 10 Pro) and $\mathcal{S}$ operates on a 64-bit Ubuntu 16.04 machine with Intel i7-7800X CPU, GeForce GTX 1080 Ti GPU, and

[2] https://github.com/davidsandberg/facenet
[3] https://github.com/qqueing/DeepSpeaker-pytorch
[4] https://github.com/euzun/justitia

Table 3.1: Usage and number of people of each dataset.

Used in	Face	Voice
DL train	MSCELEB1M (100K)	VoxCeleb (1.2K)
Fine tuning	Custom (20)	LibriSpeech (90)
Evaluation	YTF (1.6K)	LibriSpeech (831)
Security analys.	StyleGAN (60M)	

64 GB RAM. $\mathcal{C}$ and $\mathcal{S}$ have 30 megabit/s connection with 10 ms latency, and both operate on single thread. *For evaluating the prototype system, people enroll and authenticate by using different mobile devices to simulate a lost-device scenario.*

3.6 Evaluation

In this section, we evaluate the accuracy and performance of our design. We start by introducing used datasets and parameter fine tuning process. Then, we compare the accuracy and performance of our privacy-preserving protocols to the respective plaintext protocols. We also measure the impact of combining face and voice (for more security), as introduced in Sect. 3.3.2, on the accuracy and performance. On top of presenting results from whole system evaluation, to evaluate the usefulness of our QF method, we also present the results obtained *without* enabling it in Sect. 3.6.3.

3.6.1 Datasets

Table 3.1 presents the details (number of total people and usage purpose) of datasets used in this paper. In the following, we briefly explain the details of our fine-tuning and evaluation datasets.

Face

To make sure we are *fine-tuning* Justitia for realistic deployment, we collect a custom dataset of 360 facial images from 20 local subjects using the filtering and feedback mechanism through our prototype app to obtain good quality inputs.[5]

[5]This data collection activity is approved by the IRB of our organization.

In our large-scale *evaluation*, we use the YouTube Faces (YTF) academic benchmark dataset [142]. It contains noisy collections of unconstrained faces in video format belonging to 1,595 public figures. For our experiments, we extract up to 100 random frames for each identity, which is the standard protocol used to evaluate the underlying baseline DL model [24]. Note that, a subset of YTF, including faces from 312 people, pass the filters from QF layer. Hence, we measure the accuracy of our end-to-end prototype, as well as its respective plaintext baseline, using this subset for fairness (see Sect. 3.6.3). We refer this subset as "*filtered YTF*" in the rest.

Voice

We use the standard LibriSpeech (LS) dataset [143] in our voice-based pipeline. LS includes 1,000 hours of English speech recorded by 921 people in a studio environment. Since our voice samples do not contain any ambient noises, we did not apply any filter on them. We randomly pick 10% of all subjects for *fine-tuning* the parameters and use the remaining 90% of them in *evaluation*.

3.6.2 Parameters

In the following, we introduce the parameters for our prototype and our parameter selection process. In 3.2, we introduce all parameters and their fixed values for both face- and face&voice-based pipelines. *Note that, once we fix our parameters, we use them without changing across different experiments and analysis, whose results are presented in Tables 3.3, 3.4 and 3.7.*

Evaluation metrics

We fine tune and evaluate our prototype for the following two evaluation metrics: "*performance*" and "*authentication errors*".

Performance refers to the end-to-end execution times of enrollment and authentication

Table 3.2: List of parameters and their fixed values.

Par.	Description	Value
QF	pitch, yaw, roll angles	[-15, 15]
	mean val. of brightness comp. Y	[90, 120]
	quality score from [112]	[0-47.7]
$\mathcal{L}$	length of bio-bit vectors (B, B')	400 bits
ℓ	extracted bits from B for a key_i	128 bits
t	Hamming dist. thresh., $dist(B, B') \leq t$	7 bits
ϵ	max. SSF-FE Rep error probability	10^{-4}
N	number of helpers/person for $(\mathcal{L}, \ell, t, \epsilon)$	12000
R	$\mathcal{C}$'s secret, enc. in each $locker_i$ for $i \in [N]$	$\{0,1\}^{\ell}$
λ	security parameter for $0^{\lambda} \mid\mid R$	128 bits
H	hash $H = h(R)$ s.t. h is a PRF	$\{0,1\}^{\ell}$
$\mathcal{N}_{br}$	number of facial biometric readings	10
τ_{rb}	consistency threshold ratio	80%

phases through our prototype system. Note that, $\mathcal{C}$'s face scanning time is discussed as part of the usability of our system. Also, recall that this is a one-time cost that $\mathcal{C}$ pays if he use one-click authentication for day-to-day authentication, as discussed in Sect. 3.3.3.

Authentication errors refer to the standard metrics of false acceptance and false rejection rates (FAR and FRR respectively). We denote pairs *(i,j)* of biometric data from the same identity as $\mathcal{P}_{same}$ and different as $\mathcal{P}_{diff}$. To compute FAR and FRR for each pair *(i,j)* in $\mathcal{P}_{same}$, we enroll using sample i and authenticate with j, then denote the number of pairs that fail to authenticate as FR. Similarly, for each pair *(i,j)* in $\mathcal{P}_{diff}$, we enroll using sample i and authenticate with j, then denote the number of pairs that succeed to authenticate as FA. In this case, FRR= $\frac{\text{FR}}{|\mathcal{P}_{same}|}$, and FAR=$\frac{\text{FA}}{|\mathcal{P}_{diff}|}$. Our target is to have both FAR and FRR meeting industrial standard/requirement of FAR$<$0.001% and FRR$<$10% [79, 144].

Parameters of privacy-preserving layer

As discussed in Sect. 3.2.5, since the required helper data size exponentially increases with t, the amount of error we allow between bio-bit vectors (B, B') of the same person, we first limit it to be below 10 bits. Then, to guarantee the recovery of the secret R for the t-bit error tolerance, we fix SSF-FE's reproducibility error to $\epsilon = 10^{-4}$. Finally, we set the

security parameter λ and the length ℓ of the subsampled keys from (B, B'), the secret R and its hash $H = h(R)$ to 128 bits, to avoid an adversary brute-forcing the key-space of lockers.

Parameter choices for targeted errors

In the following, we summarize our parameter searching method to find the ones achieving the targeted error rates. After conducting above constraints on the fine-tuning datasets, we fix our parameters to the values presented in 3.2.

QF parameters According to our analysis conducted on our custom dataset, a facial input in high fidelity should have 1) a facial angle ranging between –15 and +15 degrees in pitch, yaw and roll, relative to the camera, 2) an average brightness value between 90 and 120 and 3) a statistical quality score below 47.7 [112].

Fine-tuning procedure Even though we fix or limit the search space of $(\ell, t, \epsilon, R, H, \lambda)$ parameters, fine-tuning them all together presents its own challenges because this is still a big search space to explore. For instance, trying all parameter combinations for our pipeline will take too long (e.g., 120 hours for single query). That is why we aim to decrease t as it plays a crucial role on overall performance, and tune other parameters step-by-step achieving minimum error rates, as followed.

- We set the number of extracted samples to $\mathcal{N}_{br} = 10$ since we target a short (e.g., 2-5 seconds) biometric capturing session.
- Then, we initially optimize $\mathcal{L}$, the length of bio-bit vectors (B, B'), extracted through DL-SE-NR layers. We set our objective to decrease the Hamming distance between B and B' for the same person while maintaining the same accuracy with the plaintext baseline system. Since $\mathcal{L}$ must be larger than ℓ, we set the search space of it from 256 to 512 bits, also as recommended by [145].

- After fixing $\mathcal{L}$, we exhaustively search for t that achieves the best accuracy. We also consider the threshold τ_{rb} for reliable bits along with these parameters. Instead of brute forcing, we follow a more probabilistic approach to find its optimal value. That is, we have to guarantee that enough bits are retained at the end of NR layer such that SSF-FE can derive N distinct masks and keys (each having ℓ subsampled bits) from B and B'.

Overall, we fix our parameters to the values presented in 3.2. With these parameters, we achieve *for the fine tuning datasets* 1) zero errors for face and 2) zero FAR at the same FRR (8.89%) for voice compared to respective plaintext pipelines on the fine-tuning datasets. For the Hamming distance threshold of $t = 7$ bits, SSF-FE Gen is responding in 0.62 sec. while SSF-FE Gen takes 0.075 sec. for the same and 0.7 sec. for different identities. Finally, we observe that more than 200 reliable bits from both face and voice-based binary hashes could be retained after filtering them over NR layer. This infers that SSF-FE can extract enough distinct masks to guarantee reproducing the secret R with the given ϵ-error probability.

3.6.3 Achieved Errors with Fixed Parameters

After fixing the parameters, as discussed in the previous section, we evaluate our protocols' error rates on the evaluation datasets of face and voice, and compare the achieved results to the error rates of respective plaintext baseline. Table 3.3 shows the results for face- and face&voice-based pipelines. It also shows the error rates, achieved when allowing $\mathcal{C}$ to try multiple authentication attempts. Recall that we achieve these results through Justitia pipeline by enabling all of its layers, but we individually discuss the impact of QF layer on the error rates after presenting our main results.

Results of face-based protocol

Results show that Justitia achieves targeted error rates, determined by industrial systems (see Sect. 3.6.2), for face-based protocol even without a requirement of multiple authenti-

Table 3.3: Accuracy comparison. Results show FRR/FAR (%) errors on YTF (face) and YTF & LS (face & voice) datasets for multiple auth. trials.

	Face			Face&Voice		
	1^{st}trial	2^{nd}	3^{rd}	1^{st}trial	2^{nd}	3^{rd}
Plaintext	0.33/0	**0/0**	0/0	8.42/0	**1.14/0**	0/0
Justitia	0.33/0	**0/0**	0/0	9.26/0	**1.32/0**	0/0

cation attempts. According to our analysis, all results for both the fine-tuning datasets and the evaluation datasets are comparable, thus *the fine-tuning process can be done once and the same parameters can be used for different deployment scenarios.*

Results of face&voice-based protocol

We now evaluate the impact of utilizing multiple biometrics, whose results also presented in Table 3.3. We randomly assign each identity from the LS dataset to an identity from the filtered YTF dataset. Even though Justitia slightly increases its FRR while using ensembled verification model, our highest priority is to have a 0% FAR since any false acceptance will be an instance of successful impersonation attack. In the following section, we will present further evaluation on the false rejection cases to show that *all users can recover from a false rejection by retrying the authentication process.*

Using multiple trials

Here, we evaluate how users of our prototype can recover from false rejections (i.e. can they successfully authenticate after a false rejection, and if so, with how many authentication attempts?). As discussed in Sect. 3.3.2, using multiple authentication trials is especially required due to the high FRR of underlying voice-verification DL. Table 3.3 shows error rates, achieved through all biometric modalities, for each number of trials. We find that 1) errors of our system are independent of the users since falsely rejected subjects change across different trials, thus retrying the authentication process will allow all legitimate users to successfully authenticate, and 2) every extra trial reduces the overall errors dramatically.

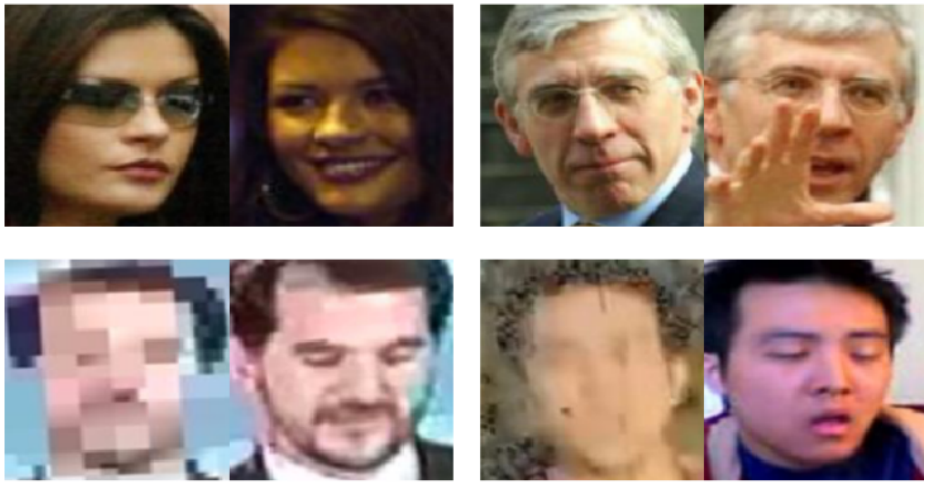


Figure 3.5: Filtered samples in QF due to being occluded, low resolution, low lit and blurry, from YouTube Faces dataset.

For instance, Justitia can achieve zero errors on face- and (1.32% FRR, 0% FAR) errors on face&voice-based protocols if we request one more authentication attempt.

Using QF layer

Using a QF layer on the client app is one of the ways to ensure high fidelity inputs to our pipeline, and thus to have similar accuracy as achieved on the fine-tuning process. As mentioned, only high quality faces of 312 subjects are retained for the enrollment/authentication tasks after applying QF to the noisy YTF dataset. Fig. 3.5 shows some of the filtered samples, eliminated due to being occluded, low resolution, blurry and low lit. Overall, if we remove the QF layer, our FRR/FAR for the YTF dataset will increase from 0.33/0% reported in Sect. 3.6.3 to 2.01/2.02%.

Comparison to plaintext baseline

In Table 3.3, we also compare achieved errors of Justitia with the respective plaintext baseline for each scenario. In plaintext pipelines, we measure how close two embeddings are to determine if they belong to the same person. Moreover, we also let the plaintext pipelines leverage from multiple samples, over a typical majority voting scheme [146], for the sake

of fair comparison. We realize this is not the exact same majority voting approach from our NR layer, but it is the best we can do for the Euclidean or cosine spaces. Other than the NR layer, we use the same settings for both schemes. For the chosen authentication attempts, Justitia achieves no further error for face-based pipeline and only an added 0.18% of FRR error for face&voice-based pipeline compared to the plaintext systems.

3.6.4 Performance of Prototype System

Now, we discuss the response time of each step in our design and networking overhead between $\mathcal{C}$ and $\mathcal{S}$ at enrollment and authentication times for face- and face&voice-based protocols. Here, we do not discuss the people's biometric sample scanning times, as it is evaluated in the following user study section. Table 3.4 presents our results. Results show that face&voice-based protocol increases the overall computation as it executes for two modalities and the client app tries multiple authentication attempts for some users, according to this protocol. Recall that, this is a one-time cost if $\mathcal{C}$ prefers to use day-to-day authentication, as discussed in Sect. 3.3.3. We compare overall response times of Justitia and prior plaintext-based protocols, which capture different biometric modalities, in the next section.

3.7 User Study on the prototype system

In this section, we first describe our user study, then we discuss the usability of our design in different scenarios, and compare it with plaintext protocols. Finally, we present a preliminary user acceptance study [6].

3.7.1 Usability

We analyze the usability of our design for both sensitive and non-sensitive authentication cases. We also discuss how combining face and voice biometrics impacts the biometric

[6]Usability examines whether our design is convenient or easy to use for desired purpose. User acceptance questions whether people would prefer using the proposed system (in which conditions).

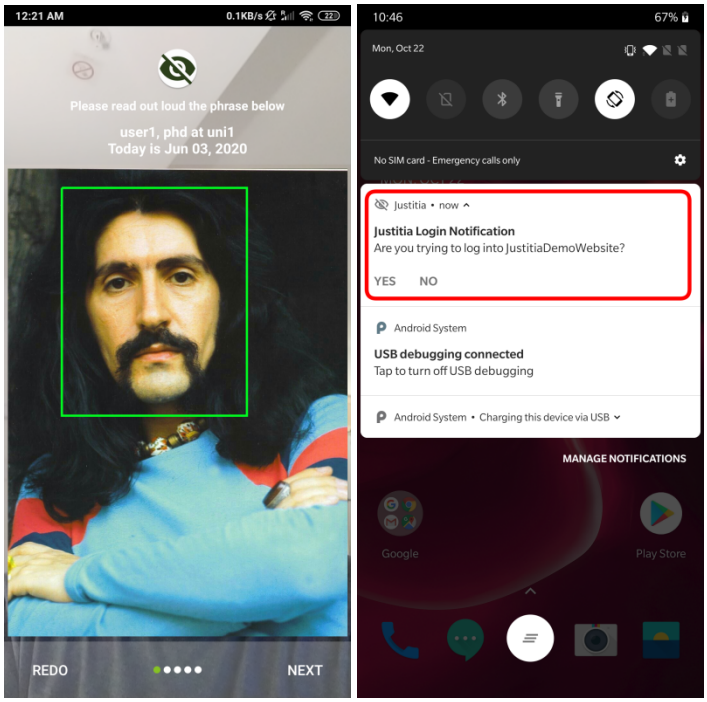


Figure 3.6: One-time enr./auth. process (left). One-click authentication message for day-to-day usage (right).

capturing process. Our user study involves 20 volunteers recruited from our organization, with the approval of the Institutional Review Board (IRB) of our organization.

Procedure

In this study, each person i) scans only his/her face and ii) scans his face while pronouncing short English phrases three times (to be used if multiple authentications will be required) both at enrollment and authentication times. Users change mobile devices and environment (e.g., background lighting) across enrollment and authentication phases to simulate a lost-device (or renewal) scenario.

Improved-usability scenario As introduced in Sect. 3.3.3, each user also iii) tries one-click authentication to log in a demo website, which simulates day-to-day usage for non-sensitive authentications. Participants try this through the above enrollment and authentication devices to make sure that biometric scanning is one-time process at this scenario. In

Table 3.4: One-time cost of enrollment/renewal. Avg. response times (sec.) of each step in the pipeline. Network also shows transferred data.

	Pha.	Cap.	QF	DL	SE	NR	FE	Network	Total
face	Enr.	2.3	0.6	0.1	0.1	0.3	0.62	0.5 sec.	4.52
	Aut.						0.08	(1.8MB)	3.98
face&voice	Enr.	11	1.2	0.4	0.5	1.2	1.49	1 sec.	16.79
	Aut.						0.19	(3.6 MB)	15.49

Fig. 3.6, we show sample screen shots from one-time enrollment/authentication and day-to-day authentication processes.

Results

Table 3.4 presents the average time it takes for the subjects in the user study to capture their i) face and ii) face&voice biometrics. Note that, some guidance messages are displayed if captured faces do not accommodate QF rules, which could increase the execution time of this process. As discussed in Sect. 3.6.4, capturing both biometrics takes longer times since the client device collects biometric samples required for multiple authentications in advance.

Results show that, participants can enroll or authenticate himself to $\mathcal{S}$ with a new (or renewed) device around 4 and 16 seconds in the first and second scenarios, respectively. Response time increases 12 seconds in the second scenario, as $\mathcal{C}$ has to record voice responses at least two times. However, this will be the case even in a plaintext-based pipeline since the voice-based deep learning baseline has a high FRR rate for single trial, as presented in Table 3.3.

Note that, these are the cost for enrollment (or device renewal) and sensitive authentications. On the other hand, as mentioned before, if $\mathcal{C}$ prefers using one-click authentication, they become a *one-time* cost before initializing a typical challenge-response authentication scheme, introduced in Sect. 3.3.3. Then, the cost of day-to-day authentications will be the same with responding an authentication message (e.g., in Fig. 3.6), which is less than a

Table 3.5: Comparing response times of Justitia to prior plaintext-based protocols for different modalities.

Protocol	Modality	Pipeline	Response Time
Justitia	Face	Private	4-4.5 sec.
Justitia	Face&voice	Private	15.5-16.8 sec.
FaceFlashing [109]	Face	Plaintext	6-29 sec.
rtCaptcha [16]	Face&voice	Plaintext	12-20 sec.

second.

Response time comparison with plaintext systems

To evaluate the impact of our privacy preserving pipeline on performance, we compare our system's response times for different modalities to the respective plaintext-based prior art. We first compare Justitia with FaceFlashing from [109], which captures only face biometrics. Then, we compare Justitia with rtCaptcha from [16], which captures both face and voice together. Notice that, both works send the captured biometric data in plaintext to $\mathcal{S}$. Table 3.5 presents the comparison results. FaceFlashing's response times increase from 6 to 29 seconds, as it increases the captured number of frames for better accuracy. Similarly, rtCaptcha has at least 12 seconds of response time, as it requires capturing voice samples for multiple trials. Results show that the privacy preserving steps in Justitia do not bring additional delay, as the bottleneck is the biometric capturing process in such authentication systems.

3.7.2 A Preliminary Study for User Acceptance

Measuring user acceptance of biometric systems, especially using a new modality like face, is a challenging task and deserves an in depth analysis. Hence, in this work, we could only present a preliminary study for our design. We refer readers to [147] for a detailed analysis reflecting the opinions of experts and non-experts on biometric authentication systems. In the following, we present the details and results of our user study to measure the user acceptance of the Justitia system.

To measure the user acceptance of Justitia relative to existing two factor authentication schemes, we conducted a large scale survey through Amazon Mechanical Turk. In particular, we recruited over 200 subjects from a broad range of genders, levels of education (from high school diploma to doctoral degree), races (Caucasian, Asian, Hispanic, and Black/African American) and ages (18 to 60). Each participant is shown a two minute video explaining what Justitia does and how it is used, *including a demo of us performing the enrollment/authentication along with the one-click authentication through our client app over different mobile devices.* After watching the video, the subjects are asked to fill out a survey containing questions designed to measure the user acceptance of Justitia and existing two factor systems. We conclude the survey with miscellaneous questions regarding their attitude towards two-factor, biometrics and passwords.

Our questions come from the widely used System Usability Scale (SUS) metric [148]. SUS presents ten opinion statements to which the subjects report how strongly they agree with each on a five-point scale ranging from "*strongly disagree*" to "*strongly agree*". We compute an overall SUS score ranging from 0 to 100 based on all the answers. A higher score indicates a *more usable* system. Based on the data collected from Mechanical Turk, we computed an average SUS score of 63 for Justitia and 66 for two factor authentication. We argue that the user acceptance of Justitia is comparable to two-factor and believe our slightly lower score is due in part to the subjects having significantly more exposure to the functionality and benefits of two-factor compared to our two minute demo video of Justitia. Given this, we consider the response to our initial prototype to be promising and believe we can improve our score with a better implementation and by giving subjects hands on experience.

We also make several observations from the responses to the miscellaneous questions. First, we notice a per-subject inverse correlation between acceptance of password managers and acceptance of our system. Second, there is a per-subject inverse correlation between trust of existing biometric systems (e.g., Touch ID) over passwords for security and ac-

Table 3.6: Comparative evaluation of various schemes through criteria set defined by Bonneau et al.[149].

	Usability								Deployability				Securiy										
Scheme	*Memorywise-Effortless*	*Scalable-for-Users*	*Nothing-to-Carry*	*Physically-Effortless*	*Easy-to-Learn*	*Efficient-to-Use*	*Infrequent-Errors*	*Easy-Recovery-from-Loss*	*Accessible*	*Negligible-Cost-per-User*	*Server-Compatible*	*Browser-Compatible*	*Resilient-to-Physical-Observation*	*Resilient-to-Targeted-Impersonation*	*Resilient-to-Throttled-Guessing*	*Resilient-to-Unthrottled-Guessing*	*Resilient-to-Internal-Observation*	*Resilient-to-Leaks-from-Other-Verifiers*	*Resilient-to-Phishing*	*Resilient-to-Theft*	*No-Trusted-Third-Party*	*Requiring-Explicit-Consent*	*Unlinkable*
Passwords			●		●	●	○	●	●	●	●	●		○						●	●	●	●
Justitia	◫	◫	○	◫	●	◫	◫	◫	⊟	●	⊟	●	◫	○	◫	◫	◫	◫	◫	⊟	●	●	●
FaceID	◫	◫	⊟	◫	●	◫	◫	⊟	⊟	⊟	⊟	⊟	◫	○	◫	◫	◫	◫	◫	●	●	⊟	●
OpenID	◫	◫	●	◫	⊟	●	◫	●	●	●	⊟	●	◫	○	◫	◫		◫		●	⊟	●	⊟
2-Step			⊟		●	⊟	⊟	⊟	⊟	⊟	⊟	●	◫	○	◫	◫		◫	◫	●	●	●	●
Iris	◫	◫	●	◫	●	⊟	⊟	⊟	⊟	⊟	⊟	⊟	◫	⊟	◫					⊟	●	⊟	⊟

●:fully offers the benefit; ○: almost offers the benefit; *no-circle*: does not offer the benefit
◫:better than passwords; ⊟: worse than passwords; *no-background*: no change

ceptance of Justitia. One possible interpretation of these results is that users who believe existing systems (be it password manager or biometric systems) are secure enough will consider Justitia to be unnecessarily complicate. We believe this shows that the security risks of existing schemes are not well understood by average users.

3.8 Comparison with Other Systems

In this section, we evaluate Justitia against the usability, deployability and security metrics proposed by Bonneau et al. [149] to demonstrate the benefits of our privacy preserving biometric authentication. Additionally, we have evaluated Apple's FaceID for comparison.

Table 3.6 presents the summary of comparative evaluation of Justitia, FaceID, OpenID, Google 2-Step Authentication (2-Step) and plaintext form of a biometric (Iris) through the criteria set defined by Bonneau et al.[149]. It should be noted that if Google 2-Step is used as single sign-on service, it does not offer security benefits of *No-Trusted-Third-Party* and *Unlinkable*. For consistency, we borrow the original work's terminology of *"fully offers"*,

"almost offers", "does not offer".

For all the usability benefits identified in [149], Justitia fully offers all but one, namely *nothing-to-carry*. For users who insist on having *nothing-to-carry*, we note that Justitia can accommodate this at the cost of being *physically-effortless*. Specifically, one can always authenticate using their biometrics on whatever client they happen to be using via recovery. Compared to FaceID, Justitia offers the additional benefit of *easy-recovery-from-loss*. This is thank to our privacy preserving pipeline, which addresses the privacy concerns limiting FaceID to local authentication. Compared to OpenID, which the related work identifies as a strong candidate to replace passwords, Justitia fully offers the benefits *memorywise-effortless* and *physically-effortless* by entirely eliminating passwords and *easy-to-learn* by guiding users through enrollment and recovery with step-by-step instructions. According to the related work, OpenID *almost* offers these benefits, but falls short because it still uses passwords. Compared to passwords in general, Justitia is better in 6 out of the 8 usability metrics and equivalent in the remaining two. Finally, the superior accuracy and privacy makes Justitia more usable than traditional biometric systems (e.g. iris).

For deployability, Justitia cannot offer the *mature* benefit since it is a prototype. Otherwise, it offers the same benefits as OpenID, mainly because it is a compliant OpenID authentication service. Furthermore, we argue that since the publication of [149], OpenID has gained further support from organizations like FIDO, Google and Facebook, thereby making server compatibility less of an issue. Overall, Justitia is the strongest in deployability among the candidates considered in Table 3.6. To the contrary, Apple's FaceID suffers greatly in this category because it is a closed ecosystem, local authentication system.

Finally, for security, Justitia in its current form does not offer the benefit of *resilient-to-targeted-attack*. In the next subsection, we discuss future work towards addressing this limitation. Also, Justitia is almost *resilient-to-theft*, but not fully because if the user's phone is stolen and has no lock screen, the thief can approve authentication attempts up to the point that the user recovers with another phone. Justitia fully offers all the other security benefits.

Table 3.7: Comparison of our security assessment technique to previous approaches. Kullback-Leibler (KL) is a pseudo-distance. First five rows are for face- and last row is for face&voice-based pipeline.

Approach	Space (data type)	Method	Security
Ours	RGB (photoreal. faces)	empiric	˜25 bits
$ZAK20_{rgb}$[84]	RGB (random)	empiric	>37 bits
$ZAK20_{emb}$[84]	Euclidean (random)	empiric	>40 bits
AYL06 [85]	KL (embeddings)	statistic	˜40 bits
D09 [86]	Hamming (face bit-vec)	statistic	˜25 bits
D09 [86]	Hamming (face&voice)	statistic	˜33 bits

On the other hand, FaceID almost offers the benefit of *resilient-to-targeted-impersonation* due to its platform security (i.e. one cannot directly feeding digital data to the iOS system without first jailbreaking it) and sophisticated sensors that supposedly detect face masks. However, there are concerns that in-person adversaries can cause victims to inadvertently unlock their phones.[7] FaceID makes this harder by checking the user's gaze, but looking at a phone is not explicit consent. Therefore, we claim FaceID almost offers *requiring-explicit-consent*. FaceID fully offers all the other security benefits.

In conclusion, comparing to all the other candidates in Table 3.6, both Justitia and FaceID offer superior security. Also, compared to password, Justitia and FaceID are superior in 6 out of 11 security criteria and inferior in 1.

3.9 Security Analysis

As pointed out in Sect. 3.4.2, our security goal is to prevent the adversary from obtaining any locked secret from the database. Recall that, even though the key space of the lockers is 128 bits, they are still vulnerable to brute force attacks if the keys are derived from the bio-bit vectors with very low entropy.

In this section, we evaluate the practical security of our design through systematic analysis by i) using existing approaches and ii) proposing a novel black-box approach.

[7] https://arstechnica.com/tech-policy/2018/10/could-border-agents-trick-you-into-unlocking-your-face-id-enabled-iphone/

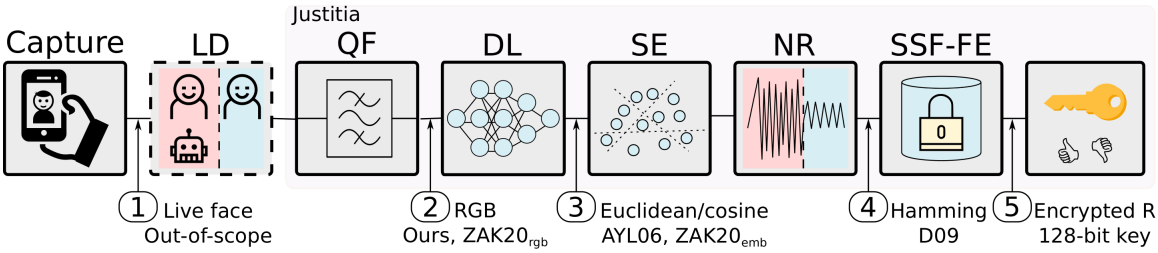


Figure 3.7: Possible attack channels, and existing (and our) security estimates through input data on each channel. LD refers to liveness detection and is out-of-scope of this work.

We also discuss the potential impact of combining face and voice on the overall security. Table 3.7 summarizes the input space (e.g., RGB, Euclidean or Hamming), assessment method (e.g., empirical or statistical), and compares the measured security (in bits, as introduced in [150]) of our approach to previous security assessment techniques, elaborated in the following. As noted before, we consider both the targeted impersonation attacks and potential defense mechanisms (e.g. liveness detection, password/pin/token protection etc.) to them as future work.

3.9.1 Categorizing the Approaches

Note that, considering the measurement data, we categorize the security assessment techniques (including ours) into two groups: "*empirical*" and "*statistical*". The former assumes an attacker probing large number of inputs, while the latter suggests fitting the limited data into a distribution model (e.g., Gaussian or binomial) to build a reflection of the real world distributions [85].

Attack channels

As depicted in Fig. 3.7, these methods could apply a simulating or make an estimation over the space of different attack channels of our design.

3.9.2 Our improvements

As presented in the following, there is no universally accepted definition or metric for security assessment of biometric authentication systems. Hence, instead of formally pinpointing the entropy of facial-biometrics, we address the limitations of the prior work (both empirical and statistical ones). We assume that the adversary can leverage from the domain knowledge while probing random inputs (as impostors) to our pipeline. This narrows down the attack space considered in [84]. Additionally, instead of conducting an analysis through intermediate steps, we simulate a black-box attack. Therefore, it already accounts for error correction and entropy loss caused by all building blocks in the pipeline. Hence, *this could be applied to different face-based authentication systems in the future.*

3.9.3 Previous Security Assessment Approaches

Now, we review the former techniques in two categories. Note that, they attack a Justitia database with people from *filtered YTF* dataset.

Empirical assessments

Zhao et al. [84] propose two approaches ($\text{ZAK20}_{\text{rgb}}$, $\text{ZAK20}_{\text{emb}}$) analyzing the security of biometric authentication systems (including face- and voice-based systems), by simulating attacks through uniformly sampled data from the space of the respective input channel. According to their simulation for facial authentication systems, an attacker can generate uniformly sampled i) raw RGB inputs and ii) embedding vectors in their message spaces. We generate and feed these inputs to the corresponding channels of our design to obtain a secret belong to any of the enrolled user.

Empirical assessment results Both $\text{ZAK20}_{\text{rgb}}$ and $\text{ZAK20}_{\text{emb}}$ attacks could not be successful even after i) *150 billion* ($> 2^{37}$) and ii) *1.6 trillion* ($> 2^{40}$) distinct input trials,

respectively. We terminate these simulations because of the limited time and another approach estimating a lower bound than these.

Statistical assessments

These approaches try to *estimate* the security of a biometric authentication system, by fitting the limited data into a distribution model that creates a reflection of the corresponding input space. In previous work, Adler et al. [85] (AYL06) uses a Gaussian model, and John Daugman [86] (D09) uses a binomial model for distribution modeling.

AYL06 estimate In this approach, the security could be interpreted as the mean of Kullback-Leibler (KL) distances of all people in a population, where KL is the relative entropy [151] between the inter- (p) and intra-person (q) biometric feature distributions. We use the embedding vectors of enrolled faces from filtered YTF to estimate p and q. Note that, intra-persons refer to face pairs from different people. Roughly speaking, AYL06 estimate KL distance of each person, by modeling p and q as Gaussian distributions. We refer readers to [85] for more details on formal definitions.

D09 estimate According to this approach, we use only intra-person distribution (among filtered YTF dataset) of subsampled keys key_i and key'_i from the bio-bit vectors B and B', respectively, for the $target$ and $imposter$ pairs. In this case, D09 suggests building a binomial model through the Hamming distances between all combinations of these genuine and impostor key pairs to realistically reflect the tails of a real world distribution, which determines the security of our system. Before delving into details of this approach in the following, we refer readers to [86] for more details.

Recall that, ℓ is the optimal value to subsample keys key_i and key'_i. For k different sets of ℓ random bits, we extract the corresponding ℓ-bit keys from all samples in both $target$ and $impostor$. Call the resulting set of keys Key(ℓ, $target$) and Key(ℓ, $impostor$) and the i-th and j-th keys in both sets Key(ℓ, $target$, i) and Key(ℓ, $impostor$, j), respectively. For

all values of (ℓ, i, j) denoting different identities, compute the Hamming distance between Key(ℓ, $target$, i) and Key(ℓ, $impostor$, j).

D09 [86] states that if we can model the Hamming distance between genuine and impostor keys as a binomial distribution with a mean μ and degrees of freedom $\mathcal{N}$, each attempt of cracking *one* locker with an impostor sample can be seen as $\mathcal{N}$ Bernoulli trials, with the probability of having the fractional Hamming distance between the two keys being $x = n/\mathcal{N}$ as: [8]

$$\text{f(x)} = \frac{\mathcal{N}!}{n!\,(\mathcal{N}-n)!}\mu^n\,(1-\mu)^{(\mathcal{N}-n)} \tag{3.1}$$

If the SSF-FE generates k lockers for each user, the chance of an attacker using a random sample from $impostor$ to decrypt the secret (i.e. unlocking one or more of the k lockers) of a user in $target$ is: $\text{F}_k(x) = 1 - [1 - f(x)]^k$.

Since the impostor key must be an exact match (x=n=0) to crack the locker, the probability of the attacker using one sample from the $impostor$ dataset to unlock one locker generated by the $target$ dataset is: $f(0) = (1-\mu)^{\mathcal{N}}$. And, the cumulative chance to unlock one locker with k keys is:

$$\text{F}_k(0) = 1 - \left[1 - (1-\mu)^{\mathcal{N}}\right]^k \tag{3.2}$$

Since impersonating enrolled users with all the other subjects from the *filtered YTF* dataset presents the attacker an advantage by the same distribution of target faces, we extract the binomial distribution through the ($target$, $impostor$) pairs from (filtered YTF, filtered YTF). Fig. 3.8 compares the empirical and ideal binomial distributions for these ($target$, $impostor$) pairs. The figure also shows the mean (μ), standard deviation (σ) and degrees of freedom ($\mathcal{N}$) for the best fit binomial distribution along with the average statistical distance (ΔP) between the empirical (PDF) and ideal ($f(x)$) distributions. Overall,

[8] Thus, n out of the $\mathcal{N}$ Bernoulli trials is a match.

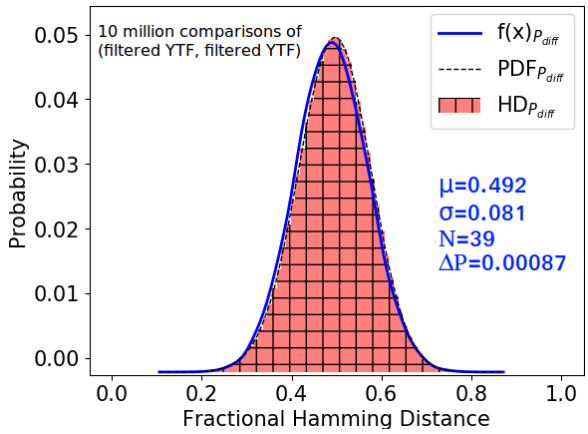


Figure 3.8: Empirical HD distributions and PDF curves (black-dashed) of 10M ($target$, $impostor$) instances and their theoretical Binomial models $f(x)$ (blue-solid and red dotted-dash).

after applying the parameters from 3.2, this approach measures the success rate of an attacker on face-based pipeline as $F_k^{-1}(0)$=1 in 24.6 million ($\tilde{}2^{25}$). Similarly, it measures the security of face&voice-based pipeline as 1 in 8 billion ($\tilde{}2^{33}$).

Statistical assessment results After applying the values derived from our data into the parameters of both approaches, AYL06 and D09, resp., compute the success rate of an attacker as around *1 in* 2^{40} and *1 in* 2^{25}. Note that, considering the applied space conversion and noise elimination operations, the gap between two estimates is not surprising. Moreover, as stated in Adler et al. [85], Gaussian models are used to give an upper bound to entropy value. In the following, we show how D09 estimation agrees with our black-box attack.

3.9.4 Proposed Security Assessment Approach

In the following, after introducing our impostor dataset, we discuss the details of our methodology and present the achieved security.

Generating impostor faces

We use the NVidia's face generator StyleGAN [152] to create unique and photorealistic faces from 60 million subjects. This simulates a powerful adversary which narrows down the random input space, considered in $\text{ZAK20}_{\text{rgb}}$, by eliminating those do not actually look like a face. We use an existing model and weights [9] trained on 70,000 high-quality images with configuration that results the lowest reported Fréchet inception distance [153], which infers generated faces will have closer image quality or more similar statistics with the real faces. We refer readers to [152] for more details regarding the diversity, quality, and stability of the generated faces.

Methodology

We measure the "security" of our system as *the chance of false acceptance when a random face sample is used to impersonate an enrolled user in our database.* This is different from the FAR reported in Sect. 3.6.3, which used a limited number of illegitimate samples as impostors.

Results

Pitted against faces from StyleGAN-generated 60 million identities, Justitia only allows 2 successful impersonations, agreeing with the statistical D09 estimate: around, 1 impersonation per 2^{25} attempts. Fig. 3.9 shows the matching $(target, impostor)$ samples from (filtered YTF, StyleGAN) pairs. Note that it is possible to generate synthetic faces with some targeted auxiliary information (e.g., gender, race, age etc.) to even increase the attack power. This observation does not undermine our conclusions.

[9] https://github.com/NVlabs/stylegan

Figure 3.9: 2 in 60M impersonations from StyleGAN, with matching YTF enrollment.

3.9.5 Security of Face&Voice-based Pipeline

If $\mathcal{C}$ combine his face and voice in a way that suggested in Sect. 3.3.2, attacker's success rate decreases to 1 impersonation per 8 billion ($\tilde{}2^{33}$) attempts. To estimate this, we use D09 as it agrees with our empirical analysis and we could not synthesize millions of voices.

3.10 Conclusion

We introduce Justitia, a privacy-preserving pipeline that authenticates $\mathcal{C}$ to a remote $\mathcal{S}$ without revealing his/her biometric data to $\mathcal{S}$. The first challenge is mapping between accurate identification component, deep learning, and privacy preserving components, like fuzzy extractors. The second one is preserving the practicality of crypto components, which is solved by our efficient and convenient noise handling approaches.

Experiments show that our design incurs zero error for face- and only a 0.18% of FRR for face&voice-based pipelines over the plaintext baseline systems to achieve privacy. We show how to increase the security and usability for sensitive and non-sensitive authentications, respectively. According to our systematical security assessments conducted through prior approaches and our novel black-box method, Justitia achieves $\tilde{}25$ bits and $\tilde{}33$ bits of security guarantees for face- and face&voice-based pipelines, respectively.

CHAPTER 4

PRIVACY PROTECTION IN BIOMETRIC SURVEILLANCE

In this chapter, I will present fuzzy labeled private set intersection (FLPSI) protocol, which is designed for protecting privacy of fuzzy data (such as biometric data) in a large scale database search.

4.1 Introduction

Recent advances in deep learning (DL)-based biometric identification have made possible real-time identification of persons in footage collected by surveillance equipment. The trend toward real-time surveillance in public and private places (e.g., streets, city halls, airports, retail stores, pharmacies, gas stations etc.) has immense benefits for public safety or customer convenience. However, adoption of these technologies come at a significant privacy cost, which raises serious objections.

To our knowledge, existing biometrics surveillance systems have the following major challenges. First, vendors store and process the collected biometric data on their server in plaintext for the ease of deployment and practicality. Second, people cannot opt-out of these systems, since video footage (or any captured faces) are directly uploaded to a remote server.

Identifying "persons of interest" may be warranted [154], but tracking *everybody else* in the process may open the doors to illegitimate surveillance and certain human right abuses [155]. In response, privacy stakeholders are pressing for a moratorium on permanent adoption of these systems, and in fact they have already succeeded in banning facial surveillance in several countries and U.S. states [20, 21, 22].

In this paper, we propose a middle ground solution, *privacy-preserving biometric search*. Here the server $\mathcal{S}$ holding a large biometric database with corresponding labels

(e.g., identities) should learn nothing about the query or the result, while the querier (client $\mathcal{C}$) should learn nothing about the database besides the label(s) of the query's match(es).

A similar problem of exact private match is extensively studied in a variety of scenarios (e.g., contact list discovery and online dating services), and can be achieved via (labeled) private set intersection (LPSI), a standard primitive [156, 157, 158, 159]. Even though the state-of-the-art CHLR18 [159] achieves a practical efficiency with communication costs sublinear to DB size, LPSI cannot directly be applied to our problem because it targets *exact* matches, while biometrics are noisy (e.g., due to lighting, imprecise scans, etc.).

We introduce FLPSI: a *fuzzy LPSI* protocol for fast privacy-preserving biometric search. We address a number of technical challenges in protocol/definition design and formal proofs.

To our knowledge, none of the prior work related to fuzzy matching achieves practical efficiency for real-time surveillance, mainly because of communication growing (at least) linearly with database size (see Sect. 4.2 and Sect. 4.11.5 for the related work). For example, two protocols of the state-of-the-art (SANNS [99]) require 1.7-5.4 GB communication and 15.1-41.7 sec. online response times over WAN per query over a million-row database.

We follow a much more scalable approach that reduces our fuzzy matching problem to an easier exact-matching subproblems that could be solved with communication cost sublinear in DB size, by leveraging optimizations of the state-of-the-art (L)PSI techniques [158, 159]. The crux of our solution is twofold. First, we translate the closeness (e.g., in Euclidean space) of two biometrics into a *t-out-of-T* set-based matching without sacrificing accuracy. That is, we encode a given biometric input into a set of T items, s.t. the two sets will likely have at least t *exactly* common items iff the biometrics are of the same person. Second, we build an efficient threshold set-matching protocol from fully homomorphic encryption (FHE), garbled circuits (GC) and *t-out-of-T* secret sharing, and solve several challenges in definitional approach.

4.1.1 Summary of Our Contributions

- We describe and formally define the functionality and security of *Fuzzy Labeled Private Set Intersection* (FLPSI). We build a FLPSI protocol using the AES blockcipher, homomorphic encryption, garbled circuits and *t-out-of-T* secret sharing. We prove the security in the semi-honest model.
- We show how to interpret closeness (e.g., in Euclidean space) between biometric inputs as *t-out-of-T* exact set-item matchings without sacrificing the accuracy.
- We give simulation-based FLPSI security definition (prior definitions of fuzzy primitives are game-based).
- We introduce a number of optimizations, in addition to the prior (L)PSI techniques we use.
- We extensively evaluate our protocol in different settings. We achieve 1.66s online running time over WAN with 40.8MB transfer per query over a million-row database.
- We systematically compare our design with prior art, and outperform all of them in their best settings, often by several orders of magnitude both in run time and communication. For example, on the largest dataset (of 10M records), we speed up by a factor of 3-33$\times$ and decrease the overall data communication by a factor of up to 48-452$\times$ compared to the two protocols of the state-of-the-art, SANNS [99].
- We highlight sublinear and concretely very small network use of our solution. In contrast with most other related work, our solution will scale on *very* small-bandwidth networks.

4.2 Related Work

As noted above, (L)PSI protocols [156, 157, 158, 159] and other exact-match protocols are inapplicable in our setting.

Freedman et al. [160] informally introduced the problem of *private fuzzy search* as an application of their private matching protocol. They proposed a basic construction, and left

improving its efficiency as future work. We now discuss state of the art techniques in fuzzy search.

4.2.1 Threshold Matching.

The works [161, 162, 163] are based on threshold *t-out-of-T* matching outlined in [160]. These constructions incur (at least) linear in DB size and concretely inefficient communication and computation. We compare them in detail in Sect. 4.11.5 and Fig. 4.12 and show that they do not scale to a million-row DB.

A related line of work uses threshold over Euclidean or Hamming distance or cosine similarity between players' vectors [164, 165, 166, 167, 168, 169, 170]. While these works are generally faster than [161, 162, 163], our solution is still orders of magnitude more efficient. We provide detailed performance comparison in Sect. 4.11.5 and Fig. 4.11.

Our solution, also based on threshold *t-out-of-T* matching, must overcome the technical difficulties of i) high variability (and hence high distance) of feature vectors of the same person, and ii) large size of extracted feature vectors (hence high costs). We resolve both by carefully integrating fine-tuned random private subsampling of the feature vectors prior to computing threshold match (see Sect. 4.5).

4.2.2 Nearest Neighbor Search (NNS).

A related line of work, albeit solving a *different* problem from the privacy perspective, is secure NNS. Indeed, NNS may (and is expected to) return match(es) for a person who is not present in DB; hence NNS does not meet our security requirements. However, we compare to NNS solutions as well, as they are *close enough in spirit* to our application scenario, and they are *faster* than prior work on private fuzzy search discussed above. Note, we do not consider outsourcing-based NNS [171, 172, 173, 174, 175] as they require a third party who learns the query ([176] requires two non-colluding servers). State-of-the-art optimized NNS [99] (SANNS) relies on a *fast* network connection (up to 56 gigabit/s) for efficiency

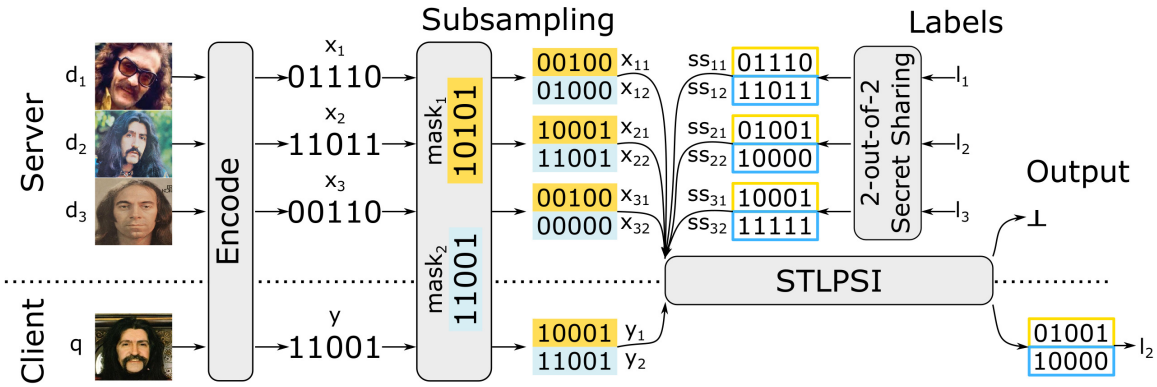


Figure 4.1: Overview of FLPSI. For clarity, subsampling is depicted without AES encryption and 2PC.

as they transfer 1.7-5.4 GB to run a 10-NNS over a database of a million entries (6.1-57.7 GB over a database of 10 million entries). Hence, SANNS is not practical enough for real-time tasks at scale. We give a detailed comparison to SANNS in Sect. 4.11.5.

Finally, we mention, but do not discuss in detail, work on fuzzy searchable encryption [177, 178], as they address a different setting where the querier owns the data stored on an untrusted server (i.e. non-private DB).

4.3 Overview and Technical Details

Here we review existing non-private (plaintext) fuzzy matching algorithms and building privacy protection into them.

4.3.1 Plaintext Fuzzy Matching

Existing facial surveillance systems, informally, work as follows. A client $\mathcal{C}$ captures facial images of people from a surveillance video footage, then transmits the biometric data to a server $\mathcal{S}$ with transport encryption, and $\mathcal{S}$ receives the data in plaintext. Then, the server uses a DL system to turn raw biometric readings into embedding vectors with a (probabilistic) guarantee that two such vectors will be close in Euclidean distance iff they are from the same person. If the server finds such a close biometric entry in its database, it returns the label (e.g, identity) of the entry to the client. Otherwise, it returns "*no match*" result to

the client. In our evaluation, we used FaceNet [24], which is the state-of-the-art DL system achieving at most 0.67% for 10 false matches per query over 1M-row DB. See Fig. 4.7.

Privacy concerns. Clearly, since the data is typically processed in plaintext by the server, it achieves maximal privacy, while the client achieves none. Next, we discuss achieving maximal client privacy as well.

4.3.2 Private Fuzzy Matching

Our goal is to build a protocol that reveals labels of query matches only to $\mathcal{C}$, while maintaining confidentiality of $\mathcal{C}$'s query and $\mathcal{S}$'s database. To achieve this, $\mathcal{C}$ and $\mathcal{S}$ can locally compute DL embeddings from their biometric data, then apply standard MPC tools to compute Euclidean (or cosine similarity) distance between the $\mathcal{C}$'s query and each of the $\mathcal{S}$'s database items [165, 169, 167, 164, 166]. However, this does not scale (cf Fig. 4.11). Our much more scalable approach is based on a *t-out-of-T matching* scheme, described in detail next. Fig. 4.1 shows a high-level overview of our FLPSI protocol.

Binary encoding. To accommodate *t-out-of-T* matching, we first address the incompatibility between DL embeddings (operating in Euclidean space) and the crypto components (operating in Hamming space) of our protocol. (Operating in Hamming space, e.g., computing closeness is *much* cheaper in MPC). To do this, parties additionally apply a space mapping function, which is based on locality-sensitive hashes [179, 145, 118, 119], to convert the embedding vectors into bio-bit vectors (x_i and y) with the desired property (they are Hamming-close if they originate from the biometrics of the same person). This is also used as a dimension reduction process in scalable image search applications [180, 181, 182]. We note that there are different DL-based algorithms that generate binary representations directly from the raw data [183, 184, 185]. We omit exploring the best algorithm, and refer to [186] for a comprehensive survey. We present our space mapping technique from [19] in Sect. 3.2.3. We will refer to the set of functions converting biometric data into bio-bit vectors, as "$Encode(.)$".

$\mathcal{C}$ and $\mathcal{S}$ proceed as follows after encoding their biometric data into bio-bit vectors y (held by $\mathcal{C}$) and $x_i \in \mathcal{X}$ (held by $\mathcal{S}$).

Subsampling: generate T random subsamples of y and each x_i bio-bit vectors (in the same way, e.g., $x_{21} = x_2 \wedge mask_1$), s.t. at least t of them will be the same iff y and a x_i belong to the same person (if bio-bit vectors are Hamming-close, this can be achieved whp);

Secret sharing: generate *t-out-of-T* secret shares of the label l_i (e.g., identity) of each $x_i \in \mathcal{X}$ (each share is associated with a subsample of x_i), such that any t shares can reconstruct l_i;

STLPSI: interactively execute a private *t-out-of-T* matching protocol (*Set Threshold LPSI, or STLPSI*) on the $\mathcal{C}$'s subsample set and the $\mathcal{S}$'s sets of (subsample, secret share) pairs[1]: the label l_i of an $x_i \in \mathcal{X}$ is revealed to $\mathcal{C}$ iff at least t of the subsamples of y and x_i are equal (which means $\mathcal{C}$ obtains shares of matching subsamples of x_i).

Our private matching achieves false positive and negative rates equal to the state-of-the-art plaintext algorithms.

4.3.3 Our Solutions to Technical Challenges

Now we discuss the most interesting technical challenges.

Subsample confidentiality As described above, $\mathcal{C}$ learns the subsamples (and respective subsampling masks), which may help $\mathcal{C}$ learn something additional about database. For example, in case of a false-positive match, the semi-honest $\mathcal{C}$ will now learn with confidence positions in bio-bit vector, thereby learning the original biometric, which may not be included in the result set. his next query so as to improve his chance of "hitting" a face in database (a false match).

We can resolve this by operating over encrypted subsamples only. For this, $\mathcal{S}$ chooses the random subsampling/projection masks and an AES encryption key $k_{\mathcal{S}}$. Then $\mathcal{S}$ via MPC

[1]Note that the secret shares are now treated as *labels* in the STLPSI.

allows $\mathcal{C}$ to compute the AES-encryptions of masked projections on the $\mathcal{C}$'s query bio-bit vector y, while keeping the projection masks and $k_{\mathcal{S}}$ secret from $\mathcal{C}$. The server efficiently computes AES-encryptions of masked projections on its large database non-interactively in $O(|\mathcal{X}|)$. Note that $\mathcal{S}$ has to refresh these masks and keys for each execution.

Concealing partial matches in single execution. (L)PSI protocols (e.g., [156, 157, 158, 159]) do not directly implement the above STLPSI functionality since they, by design, reveal partial (below-threshold t) matches. We resolve this by building effient STLPSI from *t-out-of-T* secret sharing and FHE, based on prior (L)PSI works (e.g., CHLR18 [159]).

Concealing partial matches in repeated executions. This subtle issue arises when generated shares are not refreshed between queries, and $\mathcal{C}$ may collect threshold t shares *across* queries. We resolve this by carefully resetting secret shares, subsampling masks and keys in each execution.

Novel definitional approach. In MPC, the preferred simulation-based security definitions offer clean and composable guarantees. At the same time, they require precise specification of ideal-world behavior, which we (as a community) do not know how to achieve for biometric functions. Because of this, biometric authentication definitions are usually game-based and not composable, but which allow to *bound*, rather than *precisely specify* adversary success.

One of our contributions is a *novel definitional approach* (see Sect. 4.7.1), which allows the best of both worlds: our definition is indeed simulation-based, and yet we bound adversary success rather than exactly specifying it. Our definition is generic and incorporates optional leakage, which is often needed for efficient sublinear protocols. We believe this definitional approach can serve as a template in defining primitives in the biometric space.

4.3.4 Trust Assumptions and Threat Model

- $\mathcal{C}$ and $\mathcal{S}$ locally apply binary encoding to their biometric data. We assume they own the same DL model that is trained on a public dataset and not tailored to any particu-

lar user from either party. Hence, we consider membership inference [107] or model inversion [108] attacks to be out of our scope.

- Considering our motivating application, we do not discuss here how the query biometric is obtained; we note that face detection in video footage is an easy and solved problem [187].
- We prove our 2PC (one $\mathcal{C}$ and one $\mathcal{S}$) in the semi-honest model (parties follow the protocol specification). In particular, parties do not corrupt their inputs (e.g., via a pixel perturbing attack [15]). This models natural scenarios in practice, as well as serves as a stepping stone to stronger models, such as handling malicious adversaries. Of course, many practical applications require protection against active cheating. Indeed, the biometric information served by $\mathcal{S}$ may be highly sensitive, and hence a possibility of data exfiltration by a malicious $\mathcal{C}$ would preclude offering the search service to a broader audience. We leave malicious security as important future work.

Resilience Against Certain Malicious Behaviour. While our protocol is in the SH model, we informally discuss several natural malicious attacks, their impact and mitigation.

Firstly, $\mathcal{S}$ can exclude its DB entries from search results simply by sending encryptions of random values. This can be *also* achieved by appropriate input substitution, and therefore is not an attack in a standalone execution setting. In general, efficiently ensuring that a malicious $\mathcal{S}$ is unable to omit entries in its DB is a hard technical problem.

Further, $\mathcal{C}$ can try to learn DB by querying random bit-vectors or brute-forcing arbitrary biometric inputs. Brute-forcing is a well-understood attack, which is mitigated by rate-limiting. Querying bit-vectors is not helping $\mathcal{C}$, since the bit-vector search space is larger than the space of faces.

4.4 Definition of FLPSI

In this section, we define a general syntax for a fuzzy labeled PSI. We start with the notion of closeness (fuzzy matching), adopted from [178].

Definition 4.4.1. Closeness Domain. *We say that* $\Lambda = (\mathcal{D}, \text{Cl})$ *is a* closeness domain *if* $\mathcal{D}$ *is a set, and* Cl *is the (partial)* closeness function *that takes any* $x, y \in \mathcal{D}$ *and outputs a member of* $\{\texttt{close}, \texttt{far}\}$*, so that* Cl *is symmetric (i.e.,* $\text{Cl}(x, y) = \text{Cl}(y, x)$*).*

There are no requirements on the output of `close` for pairs that are "near" (i.e. points that neither close nor far).

Definition 4.4.2. Fuzzy Labeled PSI (FLPSI). *FLPSI protocol is defined for a closeness domain* $(\mathcal{D}, \text{Cl})$ *and a label space* $\mathcal{LS}$ *by the interactive algorithm* $(\mathcal{C}, \mathcal{S})$*, where the client* $\mathcal{C}$ *inputs a query* $q \in \mathcal{D}$*, and the server* $\mathcal{S}$ *inputs a database* $Db = \{(d_1, l_1)..,(d_N, l_N)\}$*, where items* $d_i \in \mathcal{D}$ *and labels* $l_i \in \mathcal{LS}$*. At the end,* $\mathcal{C}$ *outputs a set* R*, and* $\mathcal{S}$ *outputs* $\perp$*. FLPSI must satisfy the following correctness and security properties:*

Correctness. We use ϵ-correctness instead of a perfect correctness, as it is common for biometrics-matching systems to have errors, e.g., false matches (ϵ_1) and non-matches (ϵ_2). Then, it requires that, for q and Db, we have an output set R consisting *only* of pairs (q, l_i) such that for each $i \in [N]$[2];

if $\text{Cl}(q, d_i) = \texttt{far}$, then $\Pr[(q, l_i) \notin R] \geq 1 - \epsilon_1$;

if $\text{Cl}(q, d_i) = \texttt{close}$, then $\Pr[(q, l_i) \in R] \geq 1 - \epsilon_2$, where $(d_i, l_i) \in Db$.

In our construction the domain $\mathcal{D}$ refers to facial biometrics in a surveillance scenario. Hence, $\text{Cl}(q, d_i) = \texttt{far}$ refers to each of q, d_i coming from different people, while $\text{Cl}(q, d_i) = \texttt{close}$ refers to both of them belonging to the same person. The label space $\mathcal{LS}$ refers to people's identities or other info. Hence, the client learns the information corresponding to the person in its query, if the photo(s) of this person is in the database.

Security of FLPSI is formally defined in Sect. 4.7.1 in the simulation paradigm by specifying the ideal functionality. We stress that we separately require $\Pi_{\texttt{FLPSI}}$ to satisfy the above correctness requirement. We present this low-level technical definition discussion together with the proofs in Sect. 4.7.1, and focus on the protocol description next.

[2] $[N]$ is a shorthand for $\{1, 2, \ldots, N\}$.

4.5 Building Blocks of FLPSI

In this section, we discuss the ideas behind our protocol and its building blocks (presented formally in Sect. 4.6).

4.5.1 Binary Encoding

Our construction starts with a binary encoding step, where the client and server locally turn each of their raw biometric inputs (e.g., facial photos) $q, d_i \in \mathcal{D}$, for $i \in [N]$, into bio-bit vectors $y = Encode(q)$ and $x_i = Encode(d_i)$, respectively, so that if there is a q, d_i pair of the same person, then y, x_i are likely close wrt Hamming distance.

4.5.2 Subsampling and 2PC

Now the client and server could apply random projections for each bit vector y and x_i, respectively. This outputs a set of subsampled bit vectors, $\mathcal{Y} = \{y_1, \ldots, y_T\}$ and $\mathcal{X}_i = \{x_{i1}, ..., x_{iT}\}$, with the property that if y and x_i are close, then some subsamples of them will likely be the same [73, 188, 19].

$\mathcal{S}$ hides the subsampling projections from $\mathcal{C}$ to avoid it reconstructing the inputs in the database (see Sect. 4.3.3). To do this, $\mathcal{S}$ chooses a 128-bit AES blockcipher key $k_{\mathcal{S}}$ and generates the projection masks $\{mask_1, \ldots, mask_T\}$. Note that $\mathcal{S}$ can locally compute its subsample set $\mathcal{X}_i$ for each of its bio-bit vector x_i s.t. $x_{ij} = AES_{k_{\mathcal{S}}}(x_i \wedge mask_j)$, where $j \in [T]$.

Next both parties execute a 2PC protocol $(\mathcal{C}_{AES}, \mathcal{S}_{AES})$. This protocol privately computes each $y_j \in \mathcal{Y}$ s.t. $y_j = AES_{k_{\mathcal{S}}}(y \wedge mask_j)$ for $\mathcal{C}$, s.t. it can learn matched encrypted subsamples without learning the projection masks and $k_{\mathcal{S}}$.

We implement this 2PC protocol as Yao's Garbled Circuits (GC)[3] using the EMP

[3]Note, we could have used a generic oblivious PRF (OPRF) to implement encrypted subsampling; however, known efficient OPRF are public-key based, even when both key and input are known to the evaluator. This would result in a more expensive solution, since (expensive public-key) OPRF must be applied by $\mathcal{S}$ to each DB entry for each query. In contrast, while we pay slightly more to evaluate AES inside MPC on the

toolkit [189]. We use *subsamples* and *encrypted subsamples* interchangeably, by referring $\mathcal{Y}$, throughout the paper.

4.5.3 Secret Sharing

Our construction will use STLPSI (introduced in the next section) along with a *t-out-of-T* secret sharing scheme, whose syntax, correctness and security we now define.

Definition 4.5.1. t-out-of-T Secret Sharing. *A t-out-of-T secret sharing scheme is defined by algorithms (*KS,KR*) for sharing and reconstructing a secret. The domain of secrets is* $\{0,1\}^{\mathcal{K}}$*, where* $\mathcal{K}$ *is some parameter.* KS *takes a secret* s *and outputs a set* $\{ss_1, ..., ss_T\}$ *of shares.* KR *takes as input a set of shares* $\{ss_1, ..., ss_d\}$ *and returns an integer* $s \in \{0,1\}^{\mathcal{K}}$ *if* $d \geq t$*, or* $\perp$ *if* $d < t$*.*

Correctness. For correctness we demand that for any $s \in \{0,1\}^{\mathcal{K}}$, any set $SS \in [\mathsf{KS}(s)]$, and $SS_i \subseteq SS$, where $|SS_i| \geq t$, it holds that $\mathsf{KR}(SS_i) = s$ with probability 1.

Privacy. For privacy we demand that for any $s \in \{0,1\}^{\mathcal{K}}$ and set $SS \in [\mathsf{KS}(s)]$ it holds that any subset $SS_i \subseteq SS$ of size $|SS_i| < t$ does not give any information about s, i.e., its probability distribution is independent of s.

In our protocol, $\mathcal{S}$ generates *t-out-of-T* secret shares for the label $l_i \in \mathcal{LS}$ associated with the i^{th} entry in its database. Note that $\mathcal{S}$ attaches an agreed-upon token 0^λ, where λ is a security parameter, to each label l_i before secret sharing it. Then, $\mathcal{C}$ can verify if any set of t shares (obtained via a single STLPSI execution) correctly reconstruct a valid label. Overall, for a given label $l_i \in \mathcal{LS}$, $\mathcal{S}$ generates $\{ss_{i1}, ..., ss_{iT}\} \xleftarrow{\$} \mathsf{KS}(0^\lambda \mid\mid l_i)$.

4.5.4 Set Threshold LPSI (STLPSI)

Though prior steps prepare the inputs to accommodate a *t-out-of-T* matching, existing private *t-out-of-T* matching schemes are not practical for a real-time surveillance (see

$\mathcal{C}$'s query, we pay *much* less to encrypt DB entries.

Sect. 4.11.5). We require small communication (e.g., under 128 MB), to support server with a large (1M rows) database and a WAN or less channel to the client.

A closely related LPSI primitive does achieve above performance [159], but cannot be plugged in *directly* as a building block to FLPSI, since it does not implement *t-out-of-T* matching (LPSI reveals below-threshold t matching to the client). It is, however, possible to combine with an appropriate carefully designed secret sharing scheme to achieve this feature as well.

For modularity and because STLPSI is a useful primitive, we formalize it and implement it based on prior techniques, mostly drawn from CHLR18. We integrate a number of optimizations specific to our setting, such as different bucketing, removing now redundant preprocessing steps and the use of cuckoo hashing in CHLR18. We prove security of the resulting protocol (Theorem 4.5.1).

Formal Definition of STLPSI

In this section, we formally define a general syntax and correctness for a private *t-out-of-T* matching protocol.

Definition 4.5.2. Set Threshold Labeled Private Set Intersection (STLPSI). *STLPSI protocol is defined by the input space $\mathcal{MS}$, label space $\mathcal{SS}$[a], threshold values $t, T \in \mathbb{N}$ and the interactive algorithm $(\mathcal{C}, \mathcal{S})$. $\mathcal{C}$ inputs a query set $\mathcal{Y} = \{y_1, \ldots, y_T\} \subset \mathcal{MS}$, and $\mathcal{S}$ inputs database sets $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_N\}$, where $\mathcal{X}_i = \{x_{i1}, ..., x_{iT}\} \subset \mathcal{MS}$, and $SS = \{SS_1, \ldots, SS_N\}$, where $SS_i = \{ss_{i1}, ..., ss_{iT}\} \subset \mathcal{SS}$. At the end, $\mathcal{C}$ outputs a set R, while $\mathcal{S}$ outputs $\perp$.*

[a] Since the labels are secret shares in STLPSI, we use $\mathcal{SS}$ for the label space to avoid confusion with the label space $\mathcal{LS}$ of the main protocol.

Correctness. We require that $R = \{r_1, \ldots, r_N\}$ s.t. for each $i \in [N]$, let $r'_i = \{(x_{ij}, ss_{ij}) : x_{ij} = y_j \in \mathcal{Y}, ss_{ij} \in SS_i\}_{j \in [T]}$, then we have that $r_i = (r'_i, l_i)$ iff $|r'_i| \geq t$, otherwise $r_i = \emptyset$. That is, through the set R, $\mathcal{C}$ learns such tuples (x_{ij}, ss_{ij}) and the label l_i

Functionality $\mathcal{F}_{\texttt{STLPSI}}$

Given inputs $\mathcal{Y} \subset \mathcal{MS}$ of $\mathcal{C}$, and $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_N\}$ and $SS = \{SS_1, \ldots, SS_N\}$ of $\mathcal{S}$, where $\mathcal{X}_i \subset \mathcal{MS}$ and $SS_i \subset \mathcal{SS}$, the functionality sends the output result set R (cf Def. 4.5.2) to the client, and nothing to the server.

Figure 4.2: The Ideal Functionality $\mathcal{F}_{\texttt{STLPSI}}$

associated with them iff it gets at least t exactly matching items between sets $\mathcal{Y}$ and $\mathcal{X}_i$.

Now we define the security of STLPSI. Let Π be an STLPSI protocol, defined according to Def. 4.5.2. The ideal functionality $\mathcal{F}_{\texttt{STLPSI}}$ is defined in Fig. 4.2. Next we recall the standard definition of securely realizing ideal functionality in the semi-honest model [190], formulated as Def. 4.5.3 for the 2PC case.

Definition 4.5.3. Securely Realizing Ideal Functionality. *We say that a real-world protocol Π securely realizes an ideal-world functionality $\mathcal{F}$ in the presence of static semi-honest adversaries if there exists a simulator Sim such that, for every corrupt party $P_i, i \in \{1, 2\}$ and all valid inputs x_1, x_2, the distributions $\mathrm{Real}_{\Pi}(\kappa, P_i; x_1, x_2)$ and $\mathrm{Ideal}_{\mathcal{F},\mathrm{Sim}}(\kappa, P_i; x_1, x_2)$ are computationally indistinguishable (in κ).* Real *and* Ideal *ensembles are defined as follows:*

$\mathrm{Real}_{\Pi}(\kappa, P_i; x_1, x_2)$*: run Π with security parameter κ, where each party P_i runs honestly using private input x_i. Let V_i denote the final view of party P_i, and let y_1, y_2 be the final outputs of the two parties. Output $\{V_i, (y_1, y_2)\}$.*

$\mathrm{Ideal}_{F,\mathrm{Sim}}(\kappa, P_i; x_1, x_2)$*: Let $(y_1, y_2) \leftarrow \mathcal{F}(x_1, x_2)$. Output*

$\{\mathrm{Sim}(P_i, (x_i, y_i)), (y_1, y_2)\}$.

Now we can put these together to define security of STLPSI.

Definition 4.5.4. STLPSI Security. *We say that a protocol $\Pi_{\texttt{STLPSI}} = (C, S)$, defined w.r.t. input space $\mathcal{MS}$ and label space $\mathcal{SS}$, is a secure STLPSI protocol (in the semi-honest model) with no leakage if $\Pi_{\texttt{STLPSI}}$ securely realizes (cf. Def. 4.5.3) functionality $\mathcal{F}_{\texttt{STLPSI}}$ of Fig. 4.2.*

Constructing STLPSI Protocol

For clarity, we first explain how a private *t-out-of-T* matching works on two sets, $\mathcal{Y}$ from the client and $\mathcal{X}_i \in \mathcal{X}$ from the server (each with T items) in a strawman design. Then, we introduce our optimizations for an efficient construction.

Strawman design. First, $\mathcal{C}$ and $\mathcal{S}$ agree on an FHE scheme, where $\mathcal{C}$ generates (public, secret) keys (pk, sk) and sends pk to $\mathcal{S}$. $\mathcal{C}$ also homomorphically encrypts each set item $y_j \in \mathcal{Y}$ into a ciphertext $[\![y_j]\!]$ and sends to $\mathcal{S}$. Then, $\mathcal{S}$ computes $[\![z_{ij}]\!] = r \times ([\![y_j]\!] - x_{ij}) + ss_{ij}$ under encryption[4], where $r \in_R \mathcal{P}$ and is refreshed for each computation, and ss_{ij} is the secret share (of label l_i) associated with x_{ij}. $\mathcal{S}$ sends the ciphertext $[\![z_{ij}]\!]$ to $\mathcal{C}$. Recall that secret shares are uniformly sampled items in $\mathcal{SS}$ (equal to $\mathcal{MS}$). Notice, $z_{ij} = s_{ij}$ iff $y_j = x_{ij}$, Otherwise, z_{ij} is random on $\mathcal{SS}$. Now, it is guaranteed that $\mathcal{C}$ can reconstruct the label l_i iff it gets at least t shares of l_i (see Def. 4.5.1). Otherwise, $\mathcal{C}$ learns nothing and cannot distinguish possible below-threshold t matches.

Optimizations. Applying above evaluation for each DB item does not scale to large DBs (e.g., of a million sets). We adopt the following optimizations from the (L)PSI literature for compressing DB items and reducing the circuit depth in FHE evaluations [160, 191, 192, 193, 194, 158, 157, 159]. With the exception of *bucketing*, we closely follow CHLR18 [159] in applying the following optimizations:

Polynomial interpolation is used instead of the above strawman to generate $[\![z_{ij}]\!]$. To do this, $\mathcal{S}$ interpolates an N-degree polynomial P_j, by using $(\langle \text{item} \rangle, \langle \text{secret share} \rangle)$ pairs (x_{ij}, ss_{ij}) s.t. $P_j(x_{ij}) = ss_{ij}$. Since $P_j(y) = \alpha_N y^N + ... + \alpha_1 y + \alpha_0$, where the α_i could be pre-computed by $\mathcal{S}$ in the offline phase, P_j is homomorphically evaluated in $O(\log N)$ multiplicative depth given $[\![y]\!]$. Further, a single $[\![z_{ij}]\!]$ now encodes a secret share corresponding to any of the matching x_{ij}.

Bucketing. Prior exact matching works use different methods (e.g., cuckoo hashing,

[4]Following [159], we slightly abuse notation to emphasize FHE operations with known values. Formally, we are computing $[\![z_{ij}]\!] = [\![r \times (y_j - x_{ij}) + ss_{ij}]\!]$.

bloom filters) to bucketize the DB items, so that the query item needs to be compared only with DB items in the same bucket. In our protocol, since each of T set items are generated through different LSH projections, each projection is interpreted as a bucket (with N items). Note that bucketing is a standard PSI technique [195], also used in CHLR18. It improves asymptotic performance, but concretely is costly, as buckets must be padded with dummy element for security. Crucially, this additional bucketing is *not needed* in our application. As noted above, projections already define the buckets within which the search is performed, and they need not be padded.

We combine bucketing with windowing, described next.

Windowing. Interpolating polynomials over buckets doesn't scale to large N values (e.g., a million). If $\mathcal{C}$ sends the encryptions of $y^{2^0}, y^{2^1}, y^{2^2}, ..., y^{2^{\log N}}$, $\mathcal{S}$ can homomorphically compute all necessary powers of y in $O(\log \log N)$ multiplicative depth. This technique decreases $\mathcal{C} \leftarrow \mathcal{S}$ communication cost by a factor of N, while increasing the $\mathcal{C} \rightarrow \mathcal{S}$ cost by a factor of $\log N$, which has a small impact on overall communication since $\mathcal{C}$ only holds a set of T items.

Splitting. To speed-up homomorphic evaluations, $\mathcal{S}$ splits each bucket into a partitions, s.t. it interpolates a $\frac{N}{a}$-degree polynomial per partition. This reduces the multiplicative depth to $O(\log \log \frac{N}{a})$, and the number of y powers ($\mathcal{C}$ sends to $\mathcal{S}$) to $\log \frac{N}{a}$, but increases the $\mathcal{C} \leftarrow \mathcal{S}$ communication by a factor of a as $\mathcal{S}$ sends results for all partitions to $\mathcal{C}$.

Batching. This is a well-known technique in FHE to enable Single Instruction Multiple Data (SIMD) on ciphertexts. For more details and example applications, we refer [196, 197, 193, 198, 158]. In this work, we specifically use SIMD batching from FHE library SEAL [100]. To accommodate it, $\mathcal{S}$ groups coefficients, associated with the same powers of $y_1, y_2, \ldots, y_T$ from different buckets, into vectors of length m. Since m is parameterized as $m \gg T$, $\mathcal{S}$ also concatenates coefficients from $\frac{m}{T}$ partitions. This means batching $\frac{m}{T}$ sets into a single vector, that decreases each partition size to $\frac{NT}{ma}$. Finally, $\mathcal{C}$ concatenates its set $\frac{m}{T}$ times and batches into a plaintext polynomial, then it computes all windowing

powers of it and sends encryptions of them to $\mathcal{S}$. Overall, batching decreases i) the FHE multiplicative depth to $O(\log\log\frac{NT}{ma})$; ii) the number of y powers ($\mathcal{C}$ sends to $\mathcal{S}$) to $\log\frac{NT}{ma}$; and iii) $\mathcal{C}\leftarrow\mathcal{S}$ communication by a factor of $\frac{m}{T}$.

Noise flooding. $\mathcal{S}$ re-randomizes the returned ciphertexts by adding fresh encryptions of zero with large noise [199]. This results in increased FHE parameters. See Sect. 4.11.2.

Modulus switching. This is a technique that FHE scheme allows to reduce the complexity of a ciphertext at some degrees [192]. In our design, $\mathcal{S}$ performs SEAL's modulus switching on encrypted results before sending them to $\mathcal{C}$.

After receiving the evaluation results, the client decrypts each of them to $\frac{m}{T}$ sets (each with T results). Then, it runs the reconstruction algorithm KR from Def. 4.5.1 on $\binom{T}{t}$ combinations of each set and obtains a label l_i iff at least t query subsamples match with the ones from i^{th} database set.

Full Protocol and Security Proof of STLPSI

Our STLPSI protocol and its security theorem are formally presented in Fig. 4.3 and Theorem 4.5.1, respectively. The intuition for the protocol security is presented above. In this section, we formally prove the security of our protocol $\Pi_{\texttt{STLPSI}}$ w.r.t. Def. 4.5.4.

Theorem 4.5.1. *In the presence of a semantically secure fully homomorphic encryption and t-out-of-T secret sharing schemes, the $\Pi_{\texttt{STLPSI}}$ protocol of Fig. 4.3 is a secure (in the semi-honest model) STLPSI protocol with no leakage if each of the server's input sets of labels $SS_i \in SS$ for $i \in [N]$ are:*

- *randomly sampled (and unknown to $\mathcal{C}$) t-out-of-T Shamir's secret shares of $0^\lambda \mid\mid l_i$, where λ is a security parameter and $l_i \in \mathcal{LS}$ is the label associated with i^{th} database record;*
- *the domain of secret shares $\mathcal{SS}$ is equal to the domain $\mathbb{F}_p$ of the underlying fully homomorphic encryption scheme.*

We specifically describe ideal world simulators $\mathrm{Sim}_{\mathcal{C}}$ and $\mathrm{Sim}_{\mathcal{S}}$ emulating the views

Inputs: $\mathcal{C}$ inputs set $\mathcal{Y} = \{y_1, \ldots, y_T\} \subset \mathcal{MS}$; $\mathcal{S}$ inputs $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_N\}$, where $\mathcal{X}_i = \{x_{i1}, ..., x_{iT}\} \subset \mathcal{MS}$, and $SS = \{SS_1, \ldots, SS_N\}$, where $SS_i = \{ss_{i1}, \ldots, ss_{iT}\} \subset \mathcal{SS}$ is the secret shares of $0^\lambda \mid\mid l_i$ s.t. λ is a param., $l_i \in \mathcal{LS}$.

1. **[FHE parameters]** Parties agree on FHE parameters $(m, m_p, m_{ct}, \mathcal{P})$ for an IND-CPA secure FHE scheme, and on threshold (t), split $(B = \frac{NT}{ma})$ and windowing $(w \in \{2^1, 2^2, \ldots, 2^{\log B}\})$ parameters. Then, $\mathcal{C}$ generates (public, secret) FHE keys (pk, sk), then sends pk to $\mathcal{S}$.

2. **[Pre-process $\mathcal{X}$ and SS]** $\mathcal{S}$ bucketizes $\mathcal{X}$ into a table and splits each bucket (column) into a partitions, then interpolates a B-degree polynomial P_k for each partition, s.t. $P_k(x_{ij}) = ss_{ij}$, where ss_{ij} is the secret share associated with x_{ij}. Then, $\mathcal{S}$ batches coefficients at corresponding row (of lenght m) of the table into a plaintext polynomial p_k^ℓ, where $\ell \in B$, $k \in a$.

3. **[Compute encrypted query from $\mathcal{Y}$]** $\mathcal{C}$ concatenates its input set $\frac{m}{T}$ times and batches into a plaintext polynomial. Then, it homomorphically encrypts (by using pk) each windowing power (w) of this plaintext polynomial into the ciphertexts $[\![y_w]\!]$, then sends them to $\mathcal{S}$.

4. **[Homomorphic intersection]** $\mathcal{S}$, under encryption, i) expands the received $[\![y_w]\!]$ to $\{[\![y_0]\!], [\![y_1]\!], \ldots, [\![y_B]\!]\}$; ii) evaluates $[\![z_k]\!] = \sum_{\ell=0}^{B} [\![y_\ell]\!].p_k^\ell$ for each $k \in a$, and sends all ciphertexts $[\![z_k]\!]$ to $\mathcal{C}$ after applying noise flooding and modulus switching on them.

5. **[Decrypt and get result]** $\mathcal{C}$ decrypts (by using sk) ciphertexts $[\![z_k]\!]$ to obtain result vector z_k (of length m) for each partition $k \in a$. Now, each item of z_k will be the evaluation of $P_k(y_j) = ss_{ij}$ iff there is any $x_{ij} = y_j$ in partition k of corresponding bucket, otherwise the item will be a random element in $\mathbb{F}_\mathcal{P}$.

6. **[Reconstruct label]** $\mathcal{C}$ runs KR algorithm on each $\binom{T}{t}$ combination of consecutive T items of z_k, and gets a reconstruction result s_i. We have $r_i' = \{(x_{ij}, ss_{ij}) : x_{ij} = y_j \in \mathcal{Y}, ss_{ij} \in SS_i\}_{j \in [T]}$. Then $s_i = 0^\lambda \mid\mid l_i$ and $r_i = (r_i', l_i)$ iff $|r_i'| \geq t$, otherwise s_i is a random element from $\mathbb{F}_\mathcal{P}$ (see Def. 4.5.1) and $r_i = \emptyset$. 0^λ validates a label l_i in DB.

Output: $\mathcal{C}$ outputs a result set $R = \{r_1, ..., r_N\}$, where each $r_i \neq \emptyset$ is recovered in Step 6. $\mathcal{S}$ outputs $\perp$.

Figure 4.3: STLPSI protocol $\Pi_{\mathtt{STLPSI}}$

$\mathrm{VIEW}_\mathcal{C}$ and $\mathrm{VIEW}_\mathcal{S}$ of $\mathcal{C}$ and $\mathcal{S}$ in the real execution for both protocols. Recall, the player's view includes its input, output, randomness, and the messages it received. The (challenging part of the) task of the simulators is to emulate the received messages in a

consistent manner. Recall, a simulator Sim takes as input the simulated player's input, output and leakage (if there is any). In the following, we formally present the required simulators and the proofs of the indistinguishability of the simulated and real views.

It is immediate that $\Pi_{\texttt{STLPSI}}$ correctly computes the set intersection and the associated labels if the underlying Shamir's secret reconstruction succeeds, i.e. when there are at least t intersecting items between a query and database set (cf. Def. 4.5.1). Now, we formally prove the Theorem 4.5.1.

Proof. For the ease of exposition, we assume that the simulator/protocol is parameterized by $(t, T, \lambda, m, m_p, m_{ct}, \mathcal{P}, a, B)$, which are fixed and public (see Sect. 4.11.2), and that *t-out-of-T* secret sharing scheme (Sect. 4.5.3) is used.

SIMULATING THE CLIENT. Recall, $\mathrm{Sim}_{\mathcal{C}}$ takes the client's query set $\mathcal{Y} = \{y_1, \ldots, y_T\}$ and the output of the real execution (labels of the matching database sets X_i and corresponding (item, share) pairs, if at least t of them matched). To construct $\mathrm{Sim}_{\mathcal{C}}$, we first describe the real view that needs to be simulated.

Real view of $\mathcal{C}$. In Steps 1-3 and 5-6, $\mathcal{C}$ receives no messages, and thus $\mathrm{Sim}_{\mathcal{C}}$ does nothing to simulate them.

In Step 6, $\mathcal{C}$ attempts to reconstruct a label l_i (and succeeds if there is a matching one. As required by the security of STLPSI, the client should not learn any below-threshold t matches. STLPSI achieves this since the server takes a set of secret shares (for each label) as inputs, and again, Shamir's secret sharing scheme guarantees the indistinguishability of each individual share (or any below-threshold t combinations of them) from a random item in the share domain $\mathcal{SS}$, which is the same with the agreed FHE scheme's domain $\mathbb{F}_{\mathcal{P}}$.

Also note that we assume the server randomly re-generates different set of secret shares for each label before each execution of STLPSI protocol. This prevents a serious leakage, an adversary combining (possible) below-threshold t shares, which are obtained across distinct executions, to sum up enough shares reconstructing a label.

In Step 3, $\mathcal{C}$ computes and sends $\log B$ homomorphic ciphertexts to $\mathcal{S}$. In Step 4, $\mathcal{C}$

receives back a homomorphic ciphertexts, each of which is an encryption of a degree-m FHE plaintext polynomial. Crucially, the ciphertexts sent to $\mathcal{C}$ by $\mathcal{S}$ are rerandomized with high noise (noise flooding), to hide the history of ciphertext construction.

Constructing the client's simulator. We need to simulate the output of Step 4, homomorphic evaluations of intersection functions. Hence, $\mathrm{Sim}_{\mathcal{C}}$ is defined as follows.

Recall the $\mathrm{Sim}_{\mathcal{C}}$ has the output of the real execution. Hence, if the output is empty (there is no match), $\mathrm{Sim}_{\mathcal{C}}$ generates a vectors, where each of them includes m random items from the agreed FHE scheme's domain $\mathbb{F}_{\mathcal{P}}$. And, if the output has a matching label l_i, $\mathrm{Sim}_{\mathcal{C}}$ inserts its associated shares into the corresponding vector indices (which are also obtained from the output) instead of random items from $\mathbb{F}_{\mathcal{P}}$. Then, $\mathrm{Sim}_{\mathcal{C}}$ batches each of these a vectors into a FHE plaintext polynomial and homomorphically encrypts it into a ciphertext. The ciphertext is then noise-flooded with the same noise distribution as used in the protocol. This ensures that the noise distribution in the simulated ciphertext is indistinguishable from that of the real execution. $\mathrm{Sim}_{\mathcal{C}}$ then applies modulus switching with the same parameters as in the real execution. The resulting a ciphertexts serve as a simulation of the client's view. By the IND-CPA security assumption on the agreed FHE scheme, this view is indistinguishable from the client's view $\mathrm{VIEW}_{\mathcal{C}}$ in the real execution of $\Pi_{\mathtt{STLPSI}}$.

SIMULATING THE SERVER. Simulating the server is straightforward. In Step 4, $\mathcal{S}$ receives $\log B$ ciphertexts, where each of them is an encryption of a degree-m FHE plaintext polynomial. $\mathrm{Sim}_{\mathcal{S}}$ generates new encryptions of zero in place of the encryptions in this step. By the IND-CPA security of the agreed FHE scheme, this view is indistinguishable from the server's view $\mathrm{VIEW}_{\mathcal{S}}$ in the real execution of $\Pi_{\mathtt{STLPSI}}$.

This completes the proof. □

4.6 FLPSI Protocol

In Sect. 4.3.2 we present the technical intuition of our $\Pi_{\mathtt{FLPSI}}$ protocol. Fig. 4.4 formalizes that discussion and presents $\Pi_{\mathtt{FLPSI}}$ for the closeness domain $(\mathcal{D}, \mathrm{Cl})$ and label space $\mathcal{LS}$.

The protocol uses the building blocks AES blockcipher, *t-out-of-T* secret sharing scheme $(\mathsf{KS}, \mathsf{KR})$, 2PC protocol $(\mathcal{C}_{AES}, \mathcal{S}_{AES})$, and STLPSI protocol $(\mathcal{C}_{STLPSI}, \mathcal{S}_{STLPSI})$.

In the protocol, $Encode$ is an algorithm that generates $\mathcal{L}$-bit bit vector for an input from $\mathcal{D}$; $k_{\mathcal{S}}$ is 128-bit AES blockcipher key; T is number of subsamples; t is matching threshold; λ is a security parameter; and $\wedge$ is "*logical and*" operation, used in subsampling function, i.e., $AES_{k_{\mathcal{S}}}(y \wedge mask_j)$.

The outputs of both the server's subsampling in Step 3 and $(\mathcal{C}_{AES}, \mathcal{S}_{AES})$ (Step 5), and the input items of $\mathcal{C}_{STLPSI}$ and $\mathcal{S}_{STLPSI}$ should be in the same domain $\mathcal{MS}$. Moreover, the output of secret sharing KS (Step 4) and the input labels of $\mathcal{S}_{STLPSI}$ should be from the same domain $\mathcal{SS}$.

4.6.1 Instantiating FLPSI Protocol

We now discuss specific instantiations of $\Pi_{\texttt{FLPSI}}$ building blocks, tailored for our use case. Discussion of low-level protocol and subprotocol parameters is presented in Sect. 4.11.2.

In the binary encoding step, $\mathcal{C}$ and $\mathcal{S}$ locally i) encode their raw biometrics (q and d_i, resp.) into embedding vectors, by using the state-of-the-art DL model (e.g., FaceNet [24]); and ii) translate the Euclidean space into Hamming, by using Super-Bit Locality Sensitive Hash (SBLSH) [118] (see Sect. 3.2.3), that converts DL embeddings into bio-bit vectors (y and x_i, resp.).

Next, following Sect. 4.5.2, $\mathcal{C}$ and $\mathcal{S}$ generate their subsample sets ($\mathcal{Y}$ and $\mathcal{X}_i$, resp.). Recall that, $\mathcal{S}$ generates each projection mask, by randomly choosing $\mathcal{N}_{sb}$ ones, which essentially turns all other bits into a constant zero.

Before each protocol execution, $\mathcal{S}$ regenerates the projection masks, AES encryption key $k_{\mathcal{S}}$, and secret shares of each label l_i for each DB record (by using *t-out-of-T* Shamir's secret sharing scheme [200]), so that $\mathcal{S}$ ensures that $\mathcal{C}$ is not able to use any information seen in a prior execution.

Finally, parties run STLPSI protocol in Fig. 4.3, where $\mathcal{C}$ inputs its subsamples, and

Inputs: $\mathcal{C}$ inputs query $q \in \mathcal{D}$; $\mathcal{S}$ inputs a database $Db = \{(d_1, l_1), ..., (d_N, l_N)\}$, where each $d_i \in \mathcal{D}$ and label $l_i \in \mathcal{LS}$.

1. **[Encode]** Parties agree on function $Encode : \mathcal{D} \rightarrow \{0,1\}^{\mathcal{L}}$. $\mathcal{C}$ computes $y = Encode(q)$, and $\mathcal{S}$ computes $x_i = Encode(d_i)$ for each $i \in [N]$.
2. **[Init]** The server $\mathcal{S}$ samples an AES key $k_{\mathcal{S}} \xleftarrow{\$} \{0,1\}^{128}$ and T projection masks $\{mask_1, \ldots, mask_T\}$.
3. **[Server's local subsampling]** The server generates subsample set $\mathcal{X}_i = \{x_{i1}, ..., x_{iT}\}$, where $x_{ij} \in \mathcal{MS}$ such that $x_{ij} = AES_{k_{\mathcal{S}}}(x_i \wedge mask_j)$ for each $i \in [N]$ and $j \in [T]$.
4. **[Secret sharing]** $\mathcal{S}$ generates a secret share set $SS_i = \{ss_{i1}, \ldots, ss_{iT}\} \xleftarrow{\$} \mathsf{KS}(0^\lambda \,||\, l_i)$ for each $i \in [N]$, where $ss_{ij} \in \mathcal{SS}$, 0^λ is an agreed token, and l_i is the i^{th} label.
5. **[Client's 2PC oblivious subsampling]** $\mathcal{C}$ and $\mathcal{S}$ run $(\mathcal{C}_{AES}, \mathcal{S}_{AES})$, where $\mathcal{C}_{AES}$ inputs y, $\mathcal{S}_{AES}$ inputs k_S and $\{mask_1, \ldots, mask_T\}$. Then, $\mathcal{C}_{AES}$ learns the subsample set $\mathcal{Y} = \{y_1, ..., y_T\}$, where $y_j \in \mathcal{MS}$ s.t. $y_j = AES_{k_{\mathcal{S}}}(y \wedge mask_j)$ for each $j \in [T]$, and $\mathcal{S}_{AES}$ learns $\perp$.
6. **[STLPSI execution]** $\mathcal{C}$ and $\mathcal{S}$ run $(\mathcal{C}_{STLPSI}, \mathcal{S}_{STLPSI})$, where $\mathcal{C}_{STLPSI}$ inputs $\mathcal{Y}$, and $\mathcal{S}_{STLPSI}$ inputs $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_N\}$, and $SS = \{SS_1, \ldots, SS_N\}$. At the end, $\mathcal{S}$ learns $\perp$, and $\mathcal{C}$ learns a set R as per Definition 4.5.2. I.e., we have $r'_i = \{(x_{ij}, ss_{ij}) : x_{ij} = y_j \in \mathcal{Y}, ss_{ij} \in SS_i\}_{j \in [T]}$ for each $i \in [N]$, and $r_i = (r'_i, l_i)$ iff $|r'_i| \geq t$, otherwise $r_i = \emptyset$. Then, $R = \{r_1, ..., r_N\}$.

Output: For each $r_i \in R$, $r_i \neq \emptyset$, $\mathcal{C}$ outputs l_i label recovered in Step 6. $\mathcal{S}$ outputs $\perp$.

Figure 4.4: FLPSI protocol $\Pi_{\texttt{FLPSI}}$

$\mathcal{S}$ inputs subsamples and their associated secret shares for each DB record. At the end, $\mathcal{S}$ learns nothing and $\mathcal{C}$ learns the label l_i (and matching items) of i^{th} database record iff the i^{th} record has at least t matching subsamples with the $\mathcal{C}$'s set.

Correctness. The protocol works correctly with small false matching (ϵ_1) and non-matching (ϵ_2) error probabilities. Since we can only empirically estimate these errors, we defer this analysis to Sect. 4.11.2. In summary, we target to get maximum error rates of $\epsilon_1 = 0.001$ and $\epsilon_2 = 0.01$ for the smallest DB.

4.7 Security Analysis of FLPSI Protocol

We start our analysis by formally defining security of FLPSI. We then state and prove security in Sect. 4.7.2.

4.7.1 Security Definition of FLPSI

SECURITY DEFINITION DISCUSSION. Following the common approach to modeling security of 2PC, we use the ideal-real paradigm and consider static security against a semi-honest adversary that can corrupt at most one participant.

We need to define an ideal functionality for FLPSI and what it means for a protocol to securely realize it. An ideal functionality takes inputs from the parties, computes the desired parties' outputs, and returns them to the parties, along with the corresponding leakage (if any). We require that the view of each party in real protocol's execution is indistinguishable from the view produced by the ideal-world simulator.

This is a common general approach, which, unfortunately, does not fit FLPSI and many natural biometric functionalities. The difficulty we are facing is that the ideal functionality must precisely match what happens in the real world. In particular, the parties' outputs should have the same distribution in both worlds on all inputs. In our case this would mean that the ideal functionality specifies the *exact* probability of any two close elements correctly identifying as close by the protocol, as well as the probability of far elements correctly identifying as far. This is *unrealistic* to achieve for real-world settings, such as facial biometrics we focus on.

This complication is *avoided* in game-based definitions, where no ideal functionality is defined, and hence there is no need to explicitly specify it. Indeed, security there can be defined as an adversary unable to succeed (e.g., learn something forbidden) with probability above a certain threshold. However, they will not guarantee that absolutely nothing additional is revealed, and such protocols are not freely composable – these are features of

ideal-real style definitions.

Our approach. We reconcile the yin and yang and achieve the best of both by defining the ideal FLPSI functionality via a reference to a real FLPSI protocol $\Pi_{\texttt{FLPSI}}$. Namely, we will say that ideal functionality outputs whatever the real $\Pi_{\texttt{FLPSI}}$ formally outputs. Recall, in our case (cf Def. 4.4.2), it is the set R or pairs (q, l_i). While at the first glance this may seem a tautology, this approach does provide a guarantee that nothing beyond the explicit protocol output is revealed. Now we are in a good place, since we can easily control explicit protocol output by specifying the *correctness* property. Indeed, Def. 4.4.2 requires (modulo errors bounded by ϵ) that the $\mathcal{C}$ outputs `close` labels, and in particular does not output `far` labels or any other information it may have learned from the view of execution.

In sum, the correctness requirement of Def. 4.4.2 guarantees that $\Pi_{\texttt{FLPSI}}$'s syntactic output only contains allowed records; the simulation-based component guarantees that no additional information (beyond the above output) is revealed. Put together, this makes a composable and usable security definition. We are not aware of this definitional approach being used in prior work, and believe it can be broadly useful, especially working with biometrics.

We caution the reader that the correctness requirement – and hence our definition – are not perfect. A "bad" protocol may exploit the allowed small manipulations of probability of returning each particular record and leak unauthorized information to $\mathcal{C}$. However, in FLPSI (in contrast, e.g., to MPC of approximations [201]), correctness condition is restrictive and severely limits possible leakage: we return input DB records which cannot be modified to leak. Moreover, our definition can be further tightened, e.g. by correctness requiring that a record return probability does not change if some other DB record changes. Formal exploration of this new definitional style for fuzzy MPC is an interesting future research direction.

We parameterize the security definition with a leakage profile describing leakage of information to the parties. This is common in the searchable encryption but is somewhat

Functionality $\mathcal{F}_{\texttt{FLPSI}}^{\mathcal{L},\Pi}$

Given inputs $q \in \mathcal{D}$ of C, and $Db = \{(d_1, l_1) \ldots, (d_N, l_N)\}$ of $\mathcal{S}$, where each $d_i \in \mathcal{D}, l_i \in \mathcal{LS}$,

- trusted party runs an honest execution of the protocol Π on the players' inputs and obtains the output result set R;
- return R and $\mathcal{L}_{\mathcal{C}}$ to the client;
- return $\mathcal{L}_{\mathcal{S}}$ to the server.

Figure 4.5: The Ideal Functionality $\mathcal{F}_{\texttt{FLPSI}}^{\mathcal{L},\Pi}$

novel for MPC-style definitions. Our construction will have very limited leakage to $\mathcal{C}$: a measure of closeness of $\mathcal{C}$'s input with $\mathcal{S}$'s entry it matched; no leakage in case of no match.

FORMAL FLPSI SECURITY DEFINITION. Let Π be an FLPSI protocol for a closeness domain $(\mathcal{D}, \text{Cl})$ and label space $\mathcal{LS}$, defined according to Def. 4.4.2. Let $\mathcal{L} = \{\mathcal{L}_{\mathcal{C}}, \mathcal{L}_{\mathcal{S}}\}$ be the leakage profile describing leakage to $\mathcal{C}$ and $\mathcal{S}$. The ideal functionality $\mathcal{F}_{\texttt{FLPSI}}$ is defined in Fig. 4.5. Then, the security of FLPSI is formally defined as follows.

Definition 4.7.1. FLPSI Security. *We say that a protocol $\Pi_{\texttt{FLPSI}} = (C, S)$ defined w.r.t. the closeness domain $(\mathcal{D}, \text{Cl})$ is a secure FLPSI protocol (in the semi-honest model) with leakage profile $\mathcal{L} = \{\mathcal{L}_{\mathcal{C}}, \mathcal{L}_{\mathcal{S}}\}$, if $\Pi_{\texttt{FLPSI}}$ securely realizes (cf. Def. 4.5.3) functionality $\mathcal{F}_{\texttt{FLPSI}}^{\mathcal{L},\Pi_{\texttt{FLPSI}}}$ of Fig. 4.5.*

4.7.2 Security Theorem and Proof of FLPSI

We now formally state the security theorem of FLPSI construction, instantiated with secure 2PC protocols $(\mathcal{C}_{AES}, \mathcal{S}_{AES})$ and $(\mathcal{C}_{STLPSI}, \mathcal{S}_{STLPSI})$, and Shamir's *t-out-of-T* secret sharing scheme. We also state the leakage profile of $\Pi_{\texttt{FLPSI}}$, then prove the security theorem of it in the semi-honest model.

Theorem 4.7.1. *Assume AES is a pseudorandom function (PRF). In the presence of secure (in the semi-honest model) 2PC protocols* $(\mathcal{C}_{AES}, \mathcal{S}_{AES})$ *and* $(\mathcal{C}_{STLPSI}, \mathcal{S}_{STLPSI})$*, and a* t-out-of-T *secret sharing scheme, the* $\Pi_{\texttt{FLPSI}}$ *protocol of Fig. 4.4 is a secure (in the semi-honest model) Fuzzy LPSI protocol with leakage profile* $\mathcal{L} = \{\mathcal{L}_{\mathcal{C}}, \mathcal{L}_{\mathcal{S}} = \perp\}$.

Leakage $\mathcal{L}_{\mathcal{S}}$ to $\mathcal{S}$ in $\Pi_{\texttt{FLPSI}}$. There is no leakage to $\mathcal{S}$.

Leakage $\mathcal{L}_{\mathcal{C}}$ to $\mathcal{C}$ in $\Pi_{\texttt{FLPSI}}$. $\Pi_{\texttt{FLPSI}}$ reveals to the client a measure of quality of the match with the database entry (i.e., the number of (obliviously) encrypted matching subsamples). In case of multiple matches, the client also learns which (obliviously) encrypted subsamples matched (i.e. were common) across the different matched database entries.

Recall that our initial privacy goal is to achieve *client privacy*, which is satisfied by revealing and leaking nothing to the server. Furthermore, we emphasize the leakage to the client is strictly less (and in fact much less) than the client learning the matching database entry(ies) (or its bit vector) of the server. It is easy to see that this leakage is inferred from the matching entry(ies) held by the server. Inspecting the relevant portion of the proof, it is easy to see that the $\mathrm{Sim}_{\mathcal{C}}$ actions informed by leakage can be easily performed without $\mathcal{L}_{\mathcal{C}}$, and with the knowledge of the matching database entries of the server.

In a typical scenario, where parties share all the information/data about matches (e.g., photos, name, age, etc. of a person of interest) with each other, this leakage does not have any security impact on the desired system.

Now we formally prove the security of our main protocol $\Pi_{\texttt{FLPSI}}$ w.r.t. Def. 4.7.1

Proof. In the following, we prove Theorem 4.7.1.

SIMULATING THE CLIENT. We first describe $\mathcal{C}$'s view, and then construct its simulator.

Describing the client's view. After describing $\mathrm{VIEW}_{\mathcal{C}}$, we explain what the simulator does to simulate the view and why this works. The simulator $\mathrm{Sim}_{\mathcal{C}}$'s inputs are a query $q \in \mathcal{D}$, the leakage $\mathcal{L}_{\mathcal{C}}$, and the output of the real execution (label(s) of the matched biometric(s), if any match occurred).

Let q be the client's query biometric data, and y be the output bio-bit vector, computed through DL, SBLSH and NR in the preprocessing. Only y is used in the rest of the protocol.

The preprocessing stage (Step 1) and Steps 2-4 are non-interactive and the client receives no messages. Hence, $\mathrm{Sim}_{\mathcal{C}}$ does nothing to simulate these steps.

In Step 5, $\mathcal{C}$ and $\mathcal{S}$ run MPC, where $\mathcal{C}$ inputs y, and $\mathcal{S}$ inputs $k_{\mathcal{S}}$ and $\{mask_1, \ldots, mask_T\}$. Then, $\mathcal{C}$ gets subsample set $\mathcal{Y} = \{y_1, \ldots, y_T\}$ s.t. $y_j = AES_{k_{\mathcal{S}}}(y \wedge mask_j)$. $\mathcal{C}$ receives MPC messages here, which (by the security of the underlying MPC protocol) carry no information and are simulated by the simulator guaranteed by the MPC protocol. However, the output of the MPC is something that $\mathcal{C}$ obtains in its view, and we need to simulate it.

In Step 6, the client submits the encrypted subsamples $\mathcal{Y} = \{y_1, \ldots, y_T\}$, received from Step 5, to the STLPSI protocol and gets the set of shares (and identities of the corresponding matching encrypted subsamples) as output, if there is a match (which means there are at least t matches). If there was no match, $\mathcal{C}$ receives the empty set from the STLPSI protocol. Because $\mathrm{Sim}_{\mathcal{C}}$ is given the output, it will know whether STLPSI returns empty. However, in case a match is returned in the FLPSI protocol, we do not know how many subsamples matched. We cannot simulate this (without leakage), as it depends, e.g., on how close $\mathcal{C}$'s and $\mathcal{S}$'s matching bio-bit vectors are. Thus, this information (the number of matched subsamples in case of a match) constitutes leakage $\mathcal{L}_{\mathcal{C}}$, and $\mathrm{Sim}_{\mathcal{C}}$ will use it for the simulation. We again emphasize that this leakage is strictly less (and in fact much less) than $\mathcal{C}$ learning the matching bio-bit vector (or the original biometric) of $\mathcal{S}$.

Constructing the client's simulator. We need to simulate the output and input of the STLPSI call in Step 6. STLPSI inputs from the client is the set of elements (simulated by random elements in the range of the AES function and which are further used in the simulation of the AES step, described next). STLPSI output to the client is a set of labels and allows to reconstruct the output of $\Pi_{\texttt{FLPSI}}$, together with the corresponding matched set elements. The $\Pi_{\texttt{FLPSI}}$ output (which is given to $\mathrm{Sim}_{\mathcal{C}}$ as input) indicates if there was a

match (or matches) and specifies the corresponding label(s).

If no match was achieved, $\mathrm{Sim}_{\mathcal{C}}$ sets the simulated output of STLPSI to be empty.

If there was a single match over the database, $\mathrm{Sim}_{\mathcal{C}}$ knows the label to be returned in STLPSI. It also uses leakage $\mathcal{L}_{\mathcal{C}}$ to determine how many subsamples should be returned in STLPSI. It then uses the received label l_i to generate the simulated secret shares input into STLPSI and obtained in the matched subsamples. $\mathrm{Sim}_{\mathcal{C}}$ then randomly chooses the set elements (from the AES outputs it simulated) to be the ones resulting in matches.

The case of multiple matches is handled similarly. The only interesting difference is in simulating how many subsamples should be returned for each label l_i. This is established with the help of the leakage $\mathcal{L}_{\mathcal{C}}$.

Having constructed the simulated input and output of $\Pi_{\mathtt{STLPSI}}$, $\mathrm{Sim}_{\mathcal{C}}$ uses the client-side simulator guaranteed by the security of $\Pi_{\mathtt{STLPSI}}$, to simulate the messages exchanged as part of the Step 6. Note that the input of the protocol is distributed according to the requirements of Theorem 4.5.1, and hence simulation goes through.

We need to simulate messages received in the MPC call of Step 5. The output of the MPC call is the T random elements chosen by $\mathrm{Sim}_{\mathcal{C}}$ as described above. The input to the MPC call is the client's input y, which is also given to $\mathrm{Sim}_{\mathcal{C}}$. Thus, the real-world messages generated by the MPC subprotocols called in Step 5 are simulated by running the client-side simulator provided by the MPC protocol.

This completes the description of the simulator $\mathrm{Sim}_{\mathcal{C}}$. As noted above, the discussion included in the view description and the simulator construction is a direct argument of the indistinguishability of the simulated and real views.

SIMULATING THE SERVER. Simulating $\mathcal{S}$ is significantly easier as it does not learn anything or receive any leakage in the protocol execution. The only protocol messages received by $\mathcal{S}$ are those of the calls to MPC and STLPSI in Steps 5 and 6. $\mathrm{Sim}_{\mathcal{S}}$ simulates inputs to both calls simply by following the protocol on its input, and there are no outputs to $\mathcal{S}$ in these steps. Thus, the messages received by $\mathcal{S}$ in these steps are simulated by the

corresponding server-side simulators of the MPC and STLPSI.

This completes the description of the simulator $\mathrm{Sim}_{\mathcal{S}}$. The argument of the indistinguishability of the simulated and real views is immediate.

This completes the proof.

□

4.8 Complexity Analysis of FLPSI

In this section, we present the computation and communication complexities wrt the database and query sizes. $\mathcal{C}$ holds a set of T subsamples for a single query, and $\mathcal{S}$ holds a database of N records with associated labels, each with T (subsample, share) pairs. Let a, B, m be the number of partitions, size of each partition and size of SIMD batching vector, respectively.

4.8.1 Communication complexity.

FLPSI includes two interactive protocols: 2PC subsampling $(\mathcal{C}_{AES}, \mathcal{S}_{AES})$ and STLPSI $(\mathcal{C}_{STLPSI}, \mathcal{S}_{STLPSI})$. Let β be the data transmission cost for a single $(\mathcal{C}_{AES}, \mathcal{S}_{AES})$ call, then the communication complexity for the former is $O(T\beta)$, which does not depend on the database size. Let ℓ be the length of an item in $\mathcal{MS}$ and $\mathcal{SS}$, which is equal to domain of FHE scheme $\mathbb{F}_{\mathcal{P}}$, where $\ell = \log \mathcal{P}$. Then, STLPSI has $O(a\ell + T\ell) = O(T\ell(\frac{N}{mB} + 1))$ communication complexity. Since mB could be parameterized to be (almost) equal to N (see Sect. 4.11.2), the total communication complexity is $O(T(\ell + \beta))$ (or $O(T\ell)$ considering the dominant term) in practice. This is *sublinear* relative to the database, but linear relative to the number of subsamples.

4.8.2 Computation complexity.

In the offline phase, $\mathcal{S}$ needs to interpolate $m \times a$ polynomials, each in the degree of $B = \frac{NT}{ma}$. Given that the interpolation has a $O(B^2)$ complexity, then the offline com-

plexity is $O(\frac{(NT)^2}{ma})$ [159]. *In the online phase*, $\mathcal{S}$ homomorphically evaluates a B-degree interpolation polynomial for all partitions, which has a $O(\frac{NT^2}{m^2})$ complexity. Since $T \ll m$, we have $O(\frac{NT}{m})$ FHE operations. Moreover, $\mathcal{C}$ tries $\binom{T}{t}$ combinations among plaintext results of each partition, which brings an additional $O(\binom{T}{t}\frac{ma}{T})$ share recovery cost through *plaintext* data. Note that we fix t to a small value for all of the evaluated datasets, thus the share recovery cost does not become a bottleneck in our pipeline (e.g., only 0.95% of the query time, as reported in Fig. 4.9).

4.9 Environment and Implementation Details

We use an Azure F72s_v2 instance, which has 72 virtual cores equivalent to that of 2.7 GHz Intel Xeon Platinum 8168 CPU and 144 GB of RAM each. We also have two sets of experiments: for *fast* and *slow* network connections between $\mathcal{C}$ and $\mathcal{S}$. While the former has 500 MB/s connection with 0.5 ms latency, the latter is having 40 MB/s with 34 ms latency. We use Ubuntu 18.04 in this instance. *Note that, even though, our design does not require a fast network connection or high number of threads, we use above environment for creating an identical comparison setting with the state-of-the-art* [99].

We implement our protocol on top of the homomorphic encryption library SEAL v3.5 [100], through Brakerski/Fan-Vercauteren (BFV) scheme [202]. To extract embedding vectors from facial images, we use the Python implementation of FaceNet[5] (with the Inception-Resnet-V1 architecture [203]) after aligning faces, as recommended in [204].

4.10 Optimizing FLPSI Implementation

In addition to applying optimization tricks to compress the database and reduce the homomorphic multiplication depth in STLPSI, as explained in Sect. 4.5.4, we further optimize our protocol for better performance and accuracy as follows.

[5]https://github.com/davidsandberg/facenet

4.10.1 Noise reduction (NR) in binary encoding.

Inspired by [80, 120, 19], the client can extract multiple face samples from a short surveillance video in order to perform noise removal. This can be done very seamlessly at some specific application scenarios. Since people cannot be completely in the same pose throughout a video recording, $\mathcal{C}$ can treat each individual frame in a video as a different sample. On the other hand, $\mathcal{S}$ can capture multiple samples per person more conveniently since it may have a controlled environment unlike $\mathcal{C}$.

In this optimization, both parties take bit vectors, generated in the binary encoding step (from Sect. 4.5.1) through multiple biometric readings, and majority vote over each bit. If a certain amount of them agree (e.g., at least 90 percent), they keep it. Otherwise, they cancel (zero-out) it. After eliminating noisy bits, the residual bit vector is given to the subsampling layer. We refer Sect. 3.2.4 for more details.

4.10.2 Subsample compression.

Since we use AES blockcipher with 128-bit key $k_{\mathcal{S}}$, we can compress its inputs to 128 bits to avoid multiple rounds of block-ciphering. This will reduce the online communication and computation costs of the 2PC subsampling protocol from Sect. 4.5.2. To do this, we can effectively compress a subsample as it mostly contains zero bits, e.g., only 14-out-of-256 bits are ones in our setting, as follows. We split the bio-bit vector into 128-bit of chunks, and evenly subsample each chunk (e.g., 7-out-of-128) without colliding subsampled bits across the chunks. For instance, if 8^{th} bit is subsampled in the first chunk, we do not subsample 8^{th} bit of the second chunk. Finally, we compute the bit-wise XOR of all chunks as the compressed output.

4.10.3 Optimizing STLPSI (load balancing).

We introduce a new optimization to balance the loads across the buckets in $\mathcal{S}$'s reconstructed database (see Sect. 4.5.4). We decrease the number of partitions, as argued next.

Note that a certain subsample(s) may be the same for too many DB entries, while the rest are shared by less of them. Also notice, it is not mathematically possible to build a (Lagrange or Newton) interpolation polynomial over such *(item, secret share)* pairs, where any two *items* are the same [205]. That is why $\mathcal{S}$ has to put each of the colliding subsamples into distinct partitions, and thus there is an unavoidable lower bound on the number of partitions, and accordingly, on the computation and communication costs. In STLPSI, before building the database, $\mathcal{S}$ truncates such (subsample, secret share) pairs after reaching a certain collision threshold, which balances the load of the each bucket of its constructed coefficient table. In Sect. 4.11.4, we empirically show the impact of this optimization on the overall costs.

4.11 Evaluation

In this section, we extensively evaluate our protocol, and then systematically compare it to the prior art. Note that we achieve our results by applying all optimizations.

4.11.1 Datasets

Evaluation datasets. We use a DL model that is pre-trained on the MSCeleb1M dataset, including over 8 million unconstrained facial images of around 100 thousand identities [141].

Query set. We use the YouTube Faces (YTF) benchmark dataset, that contains noisy collections of unconstrained facial videos from 1,595 public figures [142]. Since the preprocessing may use multiple biometric scans per person to generate reliable bio-bit vectors, we randomly pick (at most) ten frames each for $\mathcal{C}$ and $\mathcal{S}$ to test ϵ errors. We assume $\mathcal{C}$ always queries these 1,595 people over any size of DB in our experiments.

Database set. We generate photo-realistic synthetic faces to create large-scale databases since there is no such big public datasets. We use StyleGAN [152] to create databases of 10 thousand (Face-10K), 100 thousand (Face-100K) and one million (Face-1M) identities along with the YTF identities (with isolated samples from the query set).

Par.	Description	Value
t	matching threshold	2
T	number of subsamples	64
$\mathcal{L}$	length of bio-bit-vectors	256
τ_{rb}	consistency threshold ratio	0.9
$\mathcal{N}_{sb}$	number of subsampled bits	14
$k_{\mathcal{S}}$	$\mathcal{S}$'s key for a AES blockcipher	$\{0,1\}^{128}$
$\mathcal{P}$	prime mod. of domain $\mathbb{F}_{\mathcal{P}}$	8519681
λ	security param. for token 0^{λ}	$\lfloor \log \mathcal{P} \rfloor$=23
N	number of database entry	[10K-10M]

Figure 4.6: List of parameters and their fixed values.

Comparison datasets. For our comparative analysis, we use AT&T [206] and Deep1B [207] datasets, which are used in prior art. Note that we use these datasets in the same way as they are used in the prior art. AT&T[6] includes 400 facial images from 40 people, where 8 faces of each (320 in total) are kept as database items and 2 faces of each are queried. Deep1B contains a billion image descriptors (each 96 dimension vector), which is generated by passing images through a deep neural network [207]. We use the original query set, which includes 10 thousand data points, published by the authors[7]. And, we conduct queries over two subsets of Deep1B that consist of randomly selected one million and 10 million entries (labeled as Deep1B-1M and Deep1B-10M, respectively). We treat Deep1B descriptors as embedding vectors in our pipeline since it is not a facial dataset.

4.11.2 Parameters

In the following, we introduce the parameters and our parameter selection process. *Note that once we fix our parameters, we use them without changing across different experiments.*

ϵ-correctness errors. These refer to the errors in the ϵ-correctness of FLPSI. Recall from the Sect. 4.4 that, ϵ_1 infers the false matches, and ϵ_2 infers the false non-matches (or,

[6]https://www.kaggle.com/kasikrit/att-database-of-faces
[7]http://sites.skoltech.ru/compvision/noimi/

false rejection rate − FRR). Interpreting in our context, false matches denotes the number of different identities obtained other than the queried one, while false non-matches standing for the number of "not exist" results in response to querying existing people in the database.

In our experiments, we target to get at most 10 false matches and 1% false non-match rate for any of the database sizes, which meets accuracy requirement of the commercial systems [208, 79, 144].

Parameter choices for the targeted errors. In the following, we summarize our parameter searching method to find the ones achieving the targeted errors.

In Fig. 4.6, in addition to t and T, we enumerate and describe all parameters $(\mathcal{L}, \tau_{rb}, \mathcal{N}_{sb})$ required in DL, SBLSH and NR steps, which affect the errors. We first search the parameters for the plaintext baseline to see if we can obtain the targeted errors without enabling privacy-preserving blocks.

Tuning all parameters together has its own challenges because this is a big search space to explore. Since t and T values (especially t) is also critical for the complexity of our protocol, we set $t = 2$ and search for the minimum possible T value. To achieve this, we first tune the length of bio-bit vectors. Then, we brute force the $\mathcal{N}_{sb}$ and T by targeting to the minimum errors. We also consider the threshold τ_{rb} for the ratio of reliable bits along with these parameters. Instead of brute forcing, we follow a more probabilistic approach to find its optimal value. That is, we have to guarantee that enough bits are retained at the end of the NR layer to pick T distinct subsampling functions (each has $\mathcal{N}_{sb}$ ones). Hence, 1) the number of the remaining reliable bits ($\mathcal{N}_{rb}$) should be more than the number of subsampled bits in each subsampling function ($\mathcal{N}_{rb} > \mathcal{N}_{sb}$) and 2) $\binom{N_{rb}}{\mathcal{N}_{sb}} \geq T$ inequality should to be guaranteed. Finally, we fix our parameters to the values presented in Fig. 4.6.

Parameter choices for privacy-preserving blocks. The parameters of BFV scheme are three integers $(m_p, m_{ct}, \mathcal{P})$, where m_p is the polynomial modulus degree, m_{ct} is the ciphertext modulus and $\mathcal{P}$ is the plaintext modulus [98]. We initialize $m_p = 2^{13}$, $m_{ct} = 218$ bits and $\mathcal{P} = 8519681$ to always achieve at least a 128-bit security level as recommended

# of false matches	FRR (%) for Plaintext / FLPSI		
	Face-10K	Face-100K	Face-1M
1	2.89/2.95	2.93/2.97	2.99/3.01
2	1.62/1.65	1.86/1.95	2.13/2.18
3	1.26/1.32	1.64/1.66	1.97/2.01
4	1.06/1.14	1.39/1.42	1.55/1.56
5	0.92/1.01	1.14/1.18	1.18/1.25
6	0.81/0.85	0.94/0.97	1.06/1.12
7	0.72/0.77	0.83/0.86	0.92/0.94
8	0.56/0.59	0.74/0.79	0.87/0.92
9	0.53/0.58	0.69/0.74	0.73/0.79
10	0.51/0.56	0.58/0.63	0.67/0.75

Figure 4.7: FRRs of underlying plaintext matching system and FLPSI protocol for at most 10 false matches per query errors.

in [98]. These parameters allow us to perform a standard noise flooding operation as part of our STLPSI protocol (see Sect. 4.5.4). The LWE estimator[8] by Albrecht et al. [209] suggests $128 - 131$ bits security level for this setting. We switch the ciphertext modulus from 218 to 55 bits in the modulus-switching step to decrease the communication size from $\mathcal{S}$ to $\mathcal{C}$. For the parameters such as standard deviation error and secret key distribution we use the default values of SEAL. We set the SIMD vector size to $m = 8192$, and the size of the token 0^λ to 23 bits (at most), which is the same length of the labels of database records.

Achieved errors for the fixed parameters. After fixing the parameters, we measure the errors of end-to-end FLPSI protocol to see if it holds our ϵ-correctness requirement. Fig. 4.7 shows the FRRs per query for the targeted false-matches (at most 10 per query for any DB size). Note that these error rates have implications on the confidentiality of DB, and nothing relevant to the query data, which is the first privacy goal of our protocol. As mentioned before, revealing false matches (e.g., within industrial standards [208, 144, 79]) to the client is allowed since it is unavoidable in desired application. Having said that, though FLPSI slightly increases the FRR errors compared to underlying plaintext system (due to the reason explained in Sect. 4.10.3), it still holds the correctness for all settings.

[8]We use the commit fb7deba from https://bitbucket.org/malb/lwe-estimator/src/master/

Database	Offline		Online	Online response time (milliseconds)								
	Storage	Preprocess	comm.	Computation time with different number of threads							Best query	
	(MB)	time (s.)	(MB)	Th=1	8	16	32	64	72	Sp-up	fast	slow
Face-10K	5	0.94	12.1	523	93	68	**46**	57	56	11.4×	47	146
Face-100K	51	4.07	20.4	4457	635	376	257	241	**186**	24.0×	187	386
Face-1M	501	37.5	40.8	43956	5944	3058	1828	1647	**1355**	32.4×	1455	1655

Figure 4.8: FLPSI results (per query). The best computation times are in bold-face, and the best computation speed-ups are measured against the single-threaded results. Total response times are reported under the last two columns for fast/slow networks.

4.11.3 Costs of FLPSI

Fig. 4.8 shows experimental results of FLPSI protocol. For each database size N, it presents the storage needs and preprocessing times for the offline phase, total online communication overhead, and end-to-end online computation times for different number of threads (Th). We report total response times for the fast and slow network configurations, introduced in Sect. 4.9. For clarity, we discuss the results of **a single query over Face-1M** dataset in the following. We average over 100 queries for the FLPSI results.

Offline Preprocessing Cost of FLPSI. We run a one-time initialization phase to compile the DB from facial images. We do not include this cost in our summary tables. Our protocols refresh *t-out-of-T* secret sharings and AES blockcipher key $k_{\mathcal{S}}$ (both held by $\mathcal{S}$) per query. This is performed solely by $\mathcal{S}$ in expectation of the query. This cost is easily amortized (run concurrently) with an actively executing query, and we report it as an offline cost. In our experiments, $\mathcal{S}$ needs at most 501 MB of storage and 37.5 sec. to pre-compute and buffer a copy of constructed database of 1M entries. We include buffer reading time in the following online evaluations.

Online Communication Cost. We have a fixed (8.5 MB per query) communication cost from obliviously extracting the subsamples of a single bio-bit-vector of the client through the 2PC $(\mathcal{C}_{AES}, \mathcal{S}_{AES})$ protocol. Hence, this cost is independent from the database size. FLPSI achieves at most **40.8 MB per query** communication cost, which shows that we are not relying on the *fast* network connection for efficiency. The last two columns of Fig. 4.8 show that data communications increase from 100 to 300 ms (at most) even if we

switch from the fast to slow network connection. This is our major advantage compared to prior art (see Sect. 4.11.5). Hence, we can conclude that FLPSI is compatible with the existing network infrastructures of potential clients in the desired surveillance scenario.

Step	Party	Run time percent
Building encrypted query	$\mathcal{C}$	3.66%
Homomorphic evaluation	$\mathcal{S}$	91.6%
Decrypting query results	$\mathcal{C}$	3.79%
Extracting matches	$\mathcal{C}$	0.95%

Figure 4.9: Run time percent. of steps in a query over Face-1M.

Online Computation Cost. Even in the single-threaded execution scenario, FLPSI achieves promising performance (at most 44 seconds). Given that, since we spend most of the time for homomorphically evaluating the polynomials on the server side, as presented in Fig. 4.9, we can use multi-threading to speed up this computation. Note that setting up a powerful server could be more applicable than providing fast network connections (e.g., in gigabit scale) for every client. Using 72 threads achieves 32.4$\times$ faster computation compared to using a single thread. Moreover, since $\mathcal{S}$ concurrently evaluates partitions, which could be less than the number of threads for small databases, computation time does not decrease linearly (or increases) as $\mathcal{S}$ uses more threads.

Best end-to-end timing. In Fig. 4.8, we show the best achievable response times for each of the database sizes at the last two columns. Overall, by using multi-threading, FLPSI can privately search a single person over a database of a million people in **1.46 sec.** and **1.66 sec.** with fast and slow network connections, respectively. To the best of our knowledge, this is the fastest response time compared to prior art, with similar functionality, in a desired application scenario.

4.11.4 Impact of Load Balancing Optimization

Now we explain how we decrease the overall communication and computation costs, through the optimization from the Sect. 4.10.3. To do this, we repeat the experiments

Database	Communication		Response time (fast/slow)	
	(MB)	Saving	(seconds)	Speed up
Face-10K	72	6×	2.12/2.33	4.1×/3.7×
Face-100K	528	26×	17.8/21.7	4.0×/4.7×
Face-1M	2124	52×	189/199	4.3×/4.5×

Figure 4.10: FLPSI per query results taken *without* load balancing the server's buckets. Data communications are reduced by `saving` factors, and response times are improved by `speed up` factors with optimizations.

without applying this optimization, whose results are presented in Fig. 4.10. Then, we compare them with those in Fig. 4.8. For clarity, we only report total communication overheads and single threaded response times. To show the impact of our optimization, we also report the achieved saving factors in communication and speed ups in computation costs, by comparing optimized and non-optimized results. *Overall, we reduce the communication overheads up to 52× and speed up the response times up to 4.3/4.5× on fast/slow networks.*

4.11.5 End-to-end Comparison with Prior Art

In this section, we systematically compare FLPSI with previous *private fuzzy matching* protocols. Considering their functionality and security guarantees for our application scenario, we group prior art in two categories: i) *threshold matching* and ii) *k-nearest neighbor search*. In (i), as in our work, $\mathcal{S}$ may return empty result (depending on the ϵ_1 error) to $\mathcal{C}$ if no close entry exists in the database. In (ii), $\mathcal{S}$ always guarantees to return k database entries to $\mathcal{C}$ regardless of the query. While (ii) is a different functionality, we compare our work with protocols in both categories, as the state-of-the-art (SANNS [99]) in (ii) is also faster than protocols in (i), and is the fastest among protocols "close enough in spirit".

As discussed earlier, we do not compare with *exact matching protocols* (e.g., (L)PSI protocols from [156, 157, 158, 159]), as they do not support fuzzy matches. We solve a *much* harder problem than exact matching.

Comparison to Threshold Matching Approaches

As discussed in Sect. 4.2, prior art either a) applies thresholding to computed Euclidean (or Hamming and cosine similarity) distance [164, 165, 166, 167, 168, 169, 170], or b) runs *t-out-of-T* matching [160, 161, 162, 163] between query and database (feature) vectors. Though they satisfy the functionality requirement and security guarantees for our application scenario, none of them propose a practically applicable system for a real-time surveillance task.

Protocol	Communication		Resp. time (fast)	
	(MB)	Saving	(sec.)	Speed up
FLPSI	0.39	-	0.014	-
Yasuda et al. [170]†	9.92	25.5×	1.70	121×
Huang et al. [169]†	17.9	46.0×	6.08	434×
Osadchy et al. [168]†	35.2	90.3×	99.2	7086×
Blanton et al. [167]	2.8	7.18×	9.37	669×
Barni et al. [166]†	9.11	23.4×	16.0	1110×
Sadeghi et al. [165]	2.8	7.18×	15.5	1286×
Erkin et al. [164]	7.3	18.7×	18.0	1143×

Figure 4.11: Comparing FLPSI with existing *distance thresholding* protocols. Communication costs and response times per query over AT&T database. †Costs are scaled for AT&T database based on reported results in cited works.

Distance thresholding approaches. Fig. 4.11 compares concrete costs of FLPSI to prior work [164, 165, 166, 167, 168, 169, 170]. Note that the cited works report communication and computation costs linear in the database size. They achieve between 1.7-99.2 sec. response times and 2.8-35.2 MB network overheads per query over AT&T database.

Further, majority of them do not satisfy our ϵ-correctness requirements. We achieve **121-7086**× faster response time (14 ms. per query) and **7.18-90.3**× less communication for the *same* database, while meeting our ϵ-correctness requirements. Note that we consider *single* threaded execution for all works, but could not execute them in the exact same environment. However, since all run on similar clock speeds, our achieved speed-ups would slightly vary on the same environment.

Protocol	Communication	Computation
FLPSI	$O(\frac{NT}{mB}\ell) \approx O(T\ell)$	$O(\frac{NT}{m})$
CEC [163]	$O(N\lvert\mathbb{F}_p\rvert\ell)$	$O(N(\lvert\mathbb{F}_p\rvert+T)T'_\epsilon)$
YSPW [162]	$O(NT^2\ell)$	$O(N(poly(T)+T^2T'_\epsilon))$
CH_1 [161]	$O(NT\ell)$	$O(N(\binom{T}{t}poly(T)+TT'_\epsilon))$

Figure 4.12: Comparing FLPSI with existing *t-out-of-T* protocols that are still considered secure. Only the dominant terms are kept for all protocols. ℓ is the size of a ciphertext in the chosen encryption scheme. T'_ϵ is the time needed for all homomorphic operations in a single cycle.

t-out-of-T matching approaches. Systems [161, 162, 163] (referred as CH_1[9], YSPW, CEC, resp.) are existing, secure, *t-out-of-T* protocols. Fig. 4.12 compares asymptotic communication and computation complexity of [161, 162, 163] to our system. FLPSI behaves better both in computation and communication than CH_1, YSPW, and CEC protocols, as both of their communication and computation complexities are linear in database size. Further, computation and communication of CEC [163] are linear also with the domain size. In concrete terms, CEC reports 3GB communication for a database of 100 T-dimension vectors, where each vector item could be one of 4 distinct letters. Thus, CEC does not scale for our case (FLPSI operates in a domain with over 2^{23} integers). CH_1 [161] and YSPW [162] do not report concrete costs.

Protocol	Deep1B-1M				Deep1B-10M			
	Communication		Response time (fast/slow)		Communication		Response time (fast/slow)	
	Total	Saving	(seconds)	Speed up	Total	Saving	(seconds)	Speed up
FLPSI	40.8 MB	-	1.46/1.66	-	128 MB	-	12.7/13.5	-
SANNS-linear	5.39 GB	132×	5.79/41.7	3.97/25.1×	57.7 GB	452×	73.1/446	5.76/33.0×
SANNS-approx	1.72 GB	42×	1.70/15.1	1.16/9.09×	6.07 GB	48×	5.27/41.8	0.41/3.10×

Figure 4.13: Comparing FLPSI to two protocols of SANNS [99]. Best achieved response times are reported for fast/slow networks.

Comparison to kNNS Approaches

We emphasize that "k-nearest neighbor search" protocols solve a somewhat related, yet different problem, and do not meet the security guarantees we consider. Nevertheless, we

[9] Ye et al. [162] break the security of the second protocol from [161].

compare them to FLPSI because we wish to present a broader perspective and to illustrate that our work is more efficient not only than protocols for our exact problem, but than any prior work "close enough in spirit."

kNNS is related to FLPSI. Before discussing performance, we briefly explain the relevance of kNNS to our setting. Indeed, a protocol returning a nearest neighbor could be used to construc a (leaky) FLPSI, e.g. as follows: $\mathcal{C}$ and $\mathcal{S}$ run 1NNS. $\mathcal{C}$ obtains the output and checks if it meets the threshold of FLPSI before returning it (causing leakage to $\mathcal{C}$ if it does not). To search and return multiple matches, $\mathcal{C}$ and $\mathcal{S}$ could either proceed iteratively, increasing k by a small amount, or guess a larger k and risk higher leakage.

Performance comparison. We compare our design with Chen et al. [99]'s two protocols since, to our knowledge, they are the fastest protocols compared to all other kNNS approaches [93, 210, 211, 212], which do not use a trusted third-party in their pipelines.

[99] show (at least) 8-31$\times$ faster response times compared to optimally implemented prior art. They propose an optimized linear scan (SANNS-linear) and an approximate search (SANNS-approx) protocols, which are built upon additive homomorphic encryption, garbled circuits and oblivious read only memory, to conduct secure kNNS over large databases.

To conduct an almost identical comparison, we evaluate FLPSI on the same Azure instances with the same *fast/slow* network connections, as introduced in Sect. 4.9, and over the same image datasets: Deep1B-1M and Deep1B-10M.

Communication and computation costs. Fig. 4.13 compares total communication overheads and the best achieved response times through the fast/slow networks for the both database sizes. Due to our sublinear communication, FLPSI decreases required bandwidth by **132-452**$\times$ and **42-48**$\times$ (depending on the database size) compared to SANNS's linear and approximate protocols, respectively. This implies significant improvement in wall-clock time, especially on slower networks. In fact, SANNS outperforms FLPSI only on Deep1B-10M dataset, with *fast* network connection, and via its approximate algorithm.

For instance, the best response time of SANNS-approx protocol increases from 1.7 to 15.1 sec. as we switch the network from fast to slow connection. Similarly, SANNS-linear's performance decreases even more in the same situation, as it has more data overhead than their approximate protocol. On the other hand, FLPSI preserves its performance regardless of the network connection, as it has 128 MB of communication overhead even for a database of 10 million entries. Overall, we achieve up to **5.8/33**$\times$ and **1.2/9.1**$\times$ faster response times compared to SANNS's linear and approximate protocols, respectively, on the fast/slow networks.

4.12 Conclusions

We define FLPSI, fuzzy labeled private set intersection, and propose an efficient construction. In FLPSI, client $\mathcal{C}$ holds a biometric query and server $\mathcal{S}$ holds a labeled biometric database, where labels may be, e.g., persons' identities. In FLPSI, $\mathcal{C}$ learns the label *iff* the query is in the database, and $\mathcal{S}$ will learn nothing. Our definitional approach uniquely combines the properties of game-based and simulation-based definitions, and can be useful in other settings.

Designing an efficient protocol for FLPSI is challenging mainly due to the need to manage the noisiness of biometric data. We realize FLPSI in the semi-honest model from a blockcipher, garbled circuits, secret sharing, and fully homomorphic encryption.

FLPSI achieves *sublinear* communication cost relative to the database. Our experiments show that our solution scales well to massive datasets including up to 10 million entries. Additionally, our comparative results show that i) FLPSI achieves up to 48-452$\times$ less communication cost and ii) up to 3.1/33$\times$ faster response times compared to protocols from the state-of-the-art on a database of 10 million entries. Notably, FLPSI has a major advantage over prior art by not relying on high speed network connection for efficiency.

CHAPTER 5
CONCLUSION AND FUTURE WORK

In this thesis, I introduced three different systems to tackle with the challenges in privacy-preserving biometric authentication and recognition systems. First, I highlighted the vulnerability of existing cloud-based biometric verification/identification services even against the most crudest attacks. To address this problem, I introduced our live biometric verification system, rtCAPTCHA, to tell a real (and the actual) human and automated (large scale) impostors apart during an authentication process. Instead of playing a cat-and-mouse game with adversaries through pure ML-based approaches, rtCAPTCHA brings an unavoidable computation burden to the attackers by forcing them to solve a challenge in a standard cryptographic challenge-response protocol.

Then, I introduced our system, Justitia, a novel solution to privacy problem in the remote biometric authentication scenario. In Justitia, the server can verify the client without receiving his/her biometric data in the cleartext. In this context, I first show how to make fuzzy biometric inputs compatible with the cryptographic primitives, such as fuzzy extractor, without facing a major accuracy and performance loss. Then, I showed a novel security assessment technique, which assumes a powerful threat model and suggests a black-box model to measure the entropy of future biometrics-based authentication systems.

Finally, I proposed our fuzzy labeled private set intersection (FLPSI) protocol, which aims to protect the privacy of regular citizens in a biometric surveillance scenario. In this novel "fuzzy biometric matching" protocol, the client can search a biometric data over a sensitive database (e.g., a criminal database) without revealing the queried biometric data to the server. To do this, I first show how to convert biometric data of a person into a *t-out-of-T* exact-matching set elements. Then, I show how to build an LPSI protocol, that accepts these set elements as inputs, upon existing cryptographic primitives with proven se-

curity guarantees. I also introduced a novel security definition that provides a composable simulation-based model and, at the same time, binds the adversary success instead of precisely specifying it. Then, I show a formal proof of FLPSI based on this new definitional approach. We believe this new model creates a template for the future biometrics-based cryptographic primitives.

5.1 Future Work

Though I highlighted and solved different fundamental problems in the security and privacy of biometrics-based authentication and identification systems, there are various open problems and further challenges to address, as summarized in the following.

Dealing with noisy biometrics. Although the state-of-the-art deep learning techniques achieve high accuracies for different modalities, they still may not be enough for a security pipeline. Especially considering that the accuracy of these systems are usually measured through standard benchmark datasets, which might be different than a desired security application scenario. In our analysis and experiences, a real-world scenario usually requires dealing with more noisy settings (e.g., angled camera position, blocked faces, voice samples with noisy ambient sounds etc.). Having said that all of the proposed techniques in this dissertation are using off-the-shelf deep learning models as plug-in layers. That is, they can upgrade a new deep learning model, that deals with noisy data, in the future.

Privacy-preserving liveness detection. In rtCAPTCHA, we proposed a liveness detection system, where the server gets a client's selfie video, while she is responding random challenges. This pipeline obviously creates a privacy problem since the server receives the audio/video in cleartext. Making a privacy-preserving rtCAPTCHA is a challenging task since i) measuring the user's response time and transcribing her response on the client-side is not secure, and ii) doing these analyses on the server-side under encryption may not be applicable or feasible. The current practice is putting a high trust assumption on either (or both) client- or server-side, and thus solving the privacy issue in the liveness detection

systems is still an open and important problem.

Achieving malicious security in FLPSI. I presented a formal proof for the FLPSI in the semi-honest security model, where the parties exactly follows the protocol, but try to infer extra information through the transaction messages. This model satisfies the most of the real-life scenarios where the parties do not try to corrupt any inputs or divert from the protocol steps (e.g., two hospitals sharing data privately due to the regulations). On the other hand, malicious security guarantees a stronger model. For instance, the client may corrupt protocol inputs or leverage from the prior queries and create a sophisticated input to the FLPSI protocol to learn additional records from the database. We left malicious security to the future work.

Batching multiple people in the same FLPSI query. In the current FLPSI protocol, the client can search one person through a single query instance. However, it could be possible to query multiple people at the same time without causing additional performance and communication costs. This could benefit to a scenario where the client captures high volume of faces in a single frame from a surveillance video footage. I left studying batched-query to the future work.

Building a real-time surveillance system in the wild. Even though FLPSI presents the basic protocol and functionality, designing a full-fledged surveillance system is still a challenging task with many complications such as handling high volume of queries, combining multiple camera sources, solving recognition fatigues, privacy-preserving face tracking, answering multiple clients etc. On the other hand, FLPSI could be installed as a software plug-in to existing surveillance systems without additional hardware requirements. According to my analysis, there is no privacy-preserving biometric surveillance system on the commercial market or open-source community, and thus designing a practical system is our future work.

REFERENCES

[1] MasterCard. (2017). "MasterCard unveils 'selfie' security checks." https://www.theverge.com/2016/2/23/11098540/mastercard-facial-recognition-heartbeat-security.

[2] Uber. (2017). "Selfies and security, real-time id check." https://newsroom.uber.com/securityselfies/.

[3] Alipay. (2017). "Alibaba uses facial recognition tech for online payments." http://www.computerworld.com/article/2897117/alibaba-uses-facial-recognition-tech-for-online-payments.html.

[4] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of rgb videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2387–2395.

[5] J. Kietzmann, L. W. Lee, I. P. McCarthy, and T. C. Kietzmann, "Deepfakes: Trick or treat?" *Business Horizons*, vol. 63, no. 2, pp. 135–146, 2020.

[6] Microsoft, *Cognitive Services*, https://www.microsoft.com/cognitive-services/en-us/, 2017.

[7] Face++. (2017). https://www.faceplusplus.com/.

[8] Amazon. (2017). "Rekognition." https://aws.amazon.com/rekognition/faqs/.

[9] N. M. Duc and B. Q. Minh, "Your face is not your password face authentication bypassing lenovo–asus–toshiba," *Black Hat Briefings*, 2009.

[10] Y. Li, K. Xu, Q. Yan, Y. Li, and R. H. Deng, "Understanding osn-based facial disclosure against face authentication systems," in *Proceedings of the 9th ACM symposium on Information, computer and communications security*, ACM, 2014, pp. 413–424.

[11] I. Muslukhov. (2017). "Breaking liveness detection on Android (4.1.1)." https://www.youtube.com/watch?v=zYxphDK6s3I.

[12] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, "What makes tom hanks look like tom hanks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3952–3960.

[13] Y. Xu, T. Price, J.-M. Frahm, and F. Monrose, "Virtual u: Defeating face liveness detection by building virtual models from your public photos," in *25th USENIX*

Security Symposium (USENIX Security 16), USENIX Association, 2016, pp. 497–512.

[14] S. Saito, L. Wei, L. Hu, K. Nagano, and H. Li, "Photorealistic facial texture inference using deep neural networks," *arXiv preprint arXiv:1612.00523*, 2016.

[15] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[16] E. Uzun, S. P. Chung, I. Essa, and W. Lee, "rtCaptcha: A real-time captcha based liveness detection system," in *NDSS*, 2018.

[17] S. Ikeda, *Breach of biometrics database exposes 28 million records containing fingerprint and facial recognition data*, https://www.cpomagazine.com/cyber-security/breach-of-biometrics-database-exposes-28-million-records-containing-fingerprint-and-facial-recognition-data/, 2019.

[18] S. Miglani and M. Kumar, *India's billion-member biometric database raises privacy fears*, https://www.reuters.com/article/us-india-biometrics-idUSKCN0WI14E, 2020.

[19] E. Uzun, C. Yagemann, S. Chung, V. Kolesnikov, and W. Lee, "Cryptographic key derivation from biometric inferences for remote authentication," in *ASIACCS*, 2021.

[20] The New York Times, *The secretive company that might end privacy as we know it*, https://www.nytimes.com/2020/01/18/technology/clearview-privacy-facial-recognition.html, Jun. 2020.

[21] The Guardian, *South Wales police lose landmark facial recognition case*, https://www.theguardian.com/technology/2020/aug/11/south-wales-police-lose-landmark-facial-recognition-case, Jun. 2020.

[22] Business Insider, *Sanders, Merkley introduce bill to ban corporate facial recognition*, https://www.businessinsider.com/senate-bill-sanders-merkley-ban-corporate-facial-recognition-without-consent-2020-8.

[23] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[24] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[25] H. Gao, J. Yan, F. Cao, Z. Zhang, L. Lei, M. Tang, P. Zhang, X. Zhou, X. Wang, and J. Li, "A simple generic attack on text captchas," in *NDSS*, 2016.

[26] K. Li, F. Xu, J. Wang, Q. Dai, and Y. Liu, "A data-driven approach for facial expression synthesis in video," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, IEEE, 2012, pp. 57–64.

[27] G. Chetty and M. Wagner, "Multi-level liveness verification for face-voice biometric authentication," in *Biometric Consortium Conference, 2006 Biometrics Symposium: Special Session on Research at the*, IEEE, 2006, pp. 1–6.

[28] G. Pan, L. Sun, Z. Wu, and S. Lao, "Eyeblink-based anti-spoofing in face recognition from a generic webcamera," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, IEEE, 2007, pp. 1–8.

[29] J. Bai, T.-T. Ng, X. Gao, and Y.-Q. Shi, "Is physics-based liveness detection truly possible with a single image?" In *Circuits and systems (ISCAS), Proceedings of 2010 IEEE international symposium on*, IEEE, 2010, pp. 3425–3428.

[30] X. Tan, Y. Li, J. Liu, and L. Jiang, "Face liveness detection from a single image with sparse low rank bilinear discriminative model," *Computer Vision–ECCV 2010*, pp. 504–517, 2010.

[31] J. Määttä, A. Hadid, and M. Pietikäinen, "Face spoofing detection from single images using micro-texture analysis," in *Biometrics (IJCB), 2011 international joint conference on*, IEEE, 2011, pp. 1–7.

[32] A. da Silva Pinto, H. Pedrini, W. Schwartz, and A. Rocha, "Video-based face spoofing detection through visual rhythm analysis," in *Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on*, IEEE, 2012, pp. 221–228.

[33] S. Kim, S. Yu, K. Kim, Y. Ban, and S. Lee, "Face liveness detection using variable focusing," in *Biometrics (ICB), 2013 International Conference on*, IEEE, 2013, pp. 1–6.

[34] Z. Boulkenafet, J. Komulainen, and A. Hadid, "Face spoofing detection using colour texture analysis," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1818–1830, 2016.

[35] N. Erdogmus and S. Marcel, "Spoofing in 2d face recognition with 3d masks and anti-spoofing with kinect," in *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, IEEE, 2013, pp. 1–6.

[36] S. Liu, B. Yang, P. C. Yuen, and G. Zhao, "A 3d mask face anti-spoofing database with real world variations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 100–106.

[37] N. Kose and J.-L. Dugelay, "Reflectance analysis based countermeasure technique to detect face mask attacks," in *Digital Signal Processing (DSP), 2013 18th International Conference on*, IEEE, 2013, pp. 1–6.

[38] I. Manjani, S. Tariyal, M. Vatsa, R. Singh, and A. Majumdar, "Detecting silicone mask-based presentation attack via deep dictionary learning," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1713–1723, 2017.

[39] M. M. Chakka, A. Anjos, S. Marcel, R. Tronci, D. Muntoni, G. Fadda, M. Pili, N. Sirena, G. Murgia, M. Ristori, *et al.*, "Competition on counter measures to 2-d facial spoofing attacks," in *Biometrics (IJCB), 2011 International Joint Conference on*, IEEE, 2011, pp. 1–6.

[40] I. Chingovska, J. Yang, Z. Lei, D. Yi, S. Z. Li, O. Kahm, C. Glaser, N. Damer, A. Kuijper, A. Nouak, *et al.*, "The 2nd competition on counter measures to 2d face spoofing attacks," in *Biometrics (ICB), 2013 International Conference on*, IEEE, 2013, pp. 1–6.

[41] I. J. C. on Biometrics. (2017). "Competition on generalized face presentation attack detection in mobile authentication scenarios." https://sites.google.com/site/faceantispoofing/.

[42] M. Niemietz and J. Schwenk, "Ui redressing attacks on android devices," *Black Hat Abu Dhabi*, 2012.

[43] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 1999, pp. 187–194.

[44] J. Booth, A. Roussos, S. Zafeiriou, A. Ponniah, and D. Dunaway, "A 3d morphable model learnt from 10,000 faces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5543–5552.

[45] P. Huber, G. Hu, R. Tena, P. Mortazavian, P. Koppen, W. Christmas, M. Ratsch, and J. Kittler, "A multiresolution 3d morphable face model and fitting framework," in *Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2016.

[46] Y. Li, Y. Li, Q. Yan, H. Kong, and R. H. Deng, "Seeing your face is not enough: An inertial sensor-based liveness detection for face authentication," in *Proceedings*

of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, 2015, pp. 1558–1569.

[47] M. Kobayashi, T. Okabe, and Y. Sato, "Detecting forgery from static-scene video based on inconsistency in noise level functions," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 883–892, 2010.

[48] Y. Rao and J. Ni, "A deep learning approach to detection of splicing and copy-move forgeries in images," in *Information Forensics and Security (WIFS), 2016 IEEE International Workshop on*, IEEE, 2016, pp. 1–6.

[49] F. Peng and D.-l. Zhou, "Discriminating natural images and computer generated graphics based on the impact of cfa interpolation on the correlation of prnu," *Digital Investigation*, vol. 11, no. 2, pp. 111–119, 2014.

[50] F. Peng, D.-l. Zhou, M. Long, and X.-m. Sun, "Discrimination of natural images and computer generated graphics based on multi-fractal and regression analysis," *AEU-International Journal of Electronics and Communications*, vol. 71, pp. 72–81, 2017.

[51] R. Caldelli, I. Amerini, and A. Novi, "An analysis on attacker actions in fingerprint-copy attack in source camera identification," in *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*, IEEE, 2011, pp. 1–6.

[52] H. Gao, H. Liu, D. Yao, X. Liu, and U. Aickelin, "An audio captcha to distinguish humans from computers," in *Electronic Commerce and Security (ISECS), 2010 Third International Symposium on*, IEEE, 2010, pp. 265–269.

[53] Z. Wu and S. King, "Investigating gated recurrent networks for speech synthesis," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, IEEE, 2016, pp. 5140–5144.

[54] Z. Wu, P. Swietojanski, C. Veaux, S. Renals, and S. King, "A study of speaker adaptation for dnn-based speech synthesis," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[55] S. Shirali-Shahreza, Y. Ganjali, and R. Balakrishnan, "Verifying human users in speech-based interactions.," in *Interspeech*, 2011, pp. 1585–1588.

[56] Kairos, *Human Analytics*, https://www.kairos.com/, 2020.

[57] Z. Zhang, J. Yan, S. Liu, Z. Lei, D. Yi, and S. Z. Li, "A face antispoofing database with diverse attacks," in *Biometrics (ICB), 2012 5th IAPR international conference on*, IEEE, 2012, pp. 26–31.

[58] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilçi, M. Sahidullah, and A. Sizov, "Asvspoof 2015: The first automatic speaker verification spoofing and countermeasures challenge," *Training*, vol. 10, no. 15, p. 3750, 2015.

[59] D. Brodić, A. Amelio, and I. R. Draganov, "Response time analysis of text-based captcha by association rules," in *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, Springer, 2016, pp. 78–88.

[60] D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar, and A. I. Rudnicky, "Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, IEEE, vol. 1, 2006, pp. I–I.

[61] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," *arXiv preprint arXiv:1512.02595*, 2015.

[62] S. Mousazadeh and I. Cohen, "Voice activity detection in presence of transient noise using spectral clustering," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 6, pp. 1261–1271, 2013.

[63] E. Uzun and H. T. Sencar, "A preliminary examination technique for audio evidence to distinguish speech from non-speech using objective speech quality measures," *Speech Communication*, vol. 61, pp. 1–16, 2014.

[64] M. Van Segbroeck, A. Tsiartas, and S. Narayanan, "A robust frontend for vad: Exploiting contextual, discriminative and spectral cues of human voice.," in *INTERSPEECH*, 2013, pp. 704–708.

[65] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, "The end is nigh: Generic solving of text-based captchas.," in *WOOT*, 2014.

[66] Y. Fratantonio, C. Qian, S. P. Chung, and W. Lee, "Cloak and dagger: From two permissions to complete control of the ui feedback loop," in *Security and Privacy (SP), 2017 IEEE Symposium on*, IEEE, 2017, pp. 1041–1057.

[67] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, "Lipnet: Sentence-level lipreading," *arXiv preprint arXiv:1611.01599*, 2016.

[68] T. P. Keenan and I. FCIPS, "Hidden risks of biometric identifiers and how to avoid them," *BlackHat USA*, 2015.

[69] Apple Inc., *Face id security*, https://support.apple.com/en-us/HT208108, 2020.

[70] ——, *Touch id security*, https://support.apple.com/en-us/HT204587, 2020.

[71] Google, *Fingerprint security*, https://support.google.com/pixelphone/answer/6300638?hl=en&visit_id=637272039773000531-2694376495&rd=1, 2020.

[72] R. Chatterjee, M. S. Riazi, T. Chowdhury, E. Marasco, F. Koushanfar, and A. Juels, "Multisketches: Practical secure sketches using off-the-shelf biometric matching algorithms," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1171–1186.

[73] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *International conference on the theory and applications of cryptographic techniques*, Springer, 2004, pp. 523–540.

[74] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *Proceedings of ACM conference on Computer and communications security*, 1999.

[75] A. Juels and M. Sudan, "A fuzzy vault scheme," *Designs, Codes and Cryptography*, vol. 38, no. 2, pp. 237–257, 2006.

[76] K. Nandakumar and A. K. Jain, "Biometric template protection: Bridging the performance gap between theory and practice," *IEEE Signal Processing Magazine*, vol. 32, no. 5, pp. 88–100, 2015.

[77] D. Gafurov, P. Bours, B. Yang, and C. Busch, "Independent performance evaluation of pseudonymous identifier fingerprint verification algorithms," in *International Conference Image Analysis and Recognition*, Springer, 2013, pp. 63–71.

[78] Fingerprint Verification Competitions, *Stfv@icb2013*, https://biolab.csr.unibo.it/fvcongoing/UI/Form/ICB2013STFV.aspx, 2020.

[79] Microsoft, *Biometric requirements*, https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/windows-hello-biometric-requirements, 2020.

[80] S. Simhadri, J. Steel, and B. Fuller, "Cryptographic authentication from the iris," in *International Conference on Information Security*, Springer, 2019, pp. 465–485.

[81] E. P. Simoncelli and B. A. Olshausen, "Natural image statistics and neural representation," *Annual review of neuroscience*, vol. 24, no. 1, pp. 1193–1216, 2001.

[82] D. Balfanz, "Fido u2f implementation considerations," *FIDO Alliance Proposed Standard*, pp. 1–5, 2015.

[83] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: An end-to-end neural speaker embedding system," *arXiv:1705.02304*, 2017.

[84] B. Z. H. Zhao, H. J. Asghar, and M. A. Kaafar, "On the resilience of biometric authentication systems against random inputs," *NDSS*, 2020.

[85] A. Adler, R. Youmaran, and S. Loyka, "Towards a measure of biometric information," in *IEEE Canadian Conference on Electrical and Computer Engineering*, 2006.

[86] J. Daugman, "How iris recognition works," in *The essential guide to image processing*, Elsevier, 2009, pp. 715–739.

[87] I. Sluganovic, M. Roeschlin, K. B. Rasmussen, and I. Martinovic, "Using reflexive eye movements for fast challenge-response authentication," in *ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1056–1067.

[88] N. Karapanos, C. Marforio, C. Soriente, and S. Capkun, "Sound-proof: Usable two-factor authentication based on ambient sound," in *USENIX Security Symposium*, 2015, pp. 483–498.

[89] M. Azimpourkivi, U. Topkara, and B. Carbunar, "A secure mobile authentication alternative to biometrics," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017, pp. 28–41.

[90] ——, "Camera based two factor authentication through mobile and wearable devices," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, pp. 1–37, 2017.

[91] A. Acar, W. Liu, R. Beyah, K. Akkaya, and A. S. Uluagac, "A privacy-preserving multifactor authentication system," *Security and Privacy*, vol. 2, no. 5, e88, 2019.

[92] A. Acar, H. Aksu, A. S. Uluagac, and K. Akkaya, "A usable and robust continuous authentication framework using wearables," *IEEE Transactions on Mobile Computing*, 2020.

[93] P. Indyk and D. Woodruff, "Polylogarithmic private approximations and efficient matching," in *Theory of Cryptography Conference*, Springer, 2006, pp. 245–264.

[94] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Proceedings of the International Symposium on Privacy Enhancing Technologies*, Seattle, WA: Springer-Verlag, 2009, pp. 235–253, ISBN: 978-3-642-03167-0.

[95] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," 2010, pp. 229–244.

[96] Y. Huang, L. Malka, D. Evans, and J. Katz, "Efficient privacy-preserving biometric identification," 2011.

[97] Y. Zhang and F. Koushanfar, "Robust privacy-preserving fingerprint authentication," in *IEEE Intern. Symposium on Hardware Oriented Security and Trust*, 2016.

[98] M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, J. Hoffstein, K. Lauter, S. Lokam, D. Moody, T. Morrison, *et al.*, "Security of homomorphic encryption," *HomomorphicEncryption. org, Redmond WA, Tech. Rep*, 2017.

[99] H. Chen, I. Chillotti, Y. Dong, O. Poburinnaya, I. Razenshteyn, and M. S. Riazi, "SANNS: Scaling up secure approximate k-nearest neighbors search," in *29th USENIX Security Symposium*, Boston, MA, Aug. 2020.

[100] *Microsoft SEAL (release 3.5)*, https://github.com/Microsoft/SEAL, Microsoft Research, Redmond, WA., Aug. 2020.

[101] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys*, vol. 51, no. 4, 2018.

[102] M. Gomez-Barrero, E. Maiorana, J. Galbally, P. Campisi, and J. Fierrez, "Multi-biometric template protection based on homomorphic encryption," *Pattern Recognition*, vol. 67, pp. 149–163, 2017.

[103] M. Gomez-Barrero, J. Fierrez, J. Galbally, E. Maiorana, and P. Campisi, "Implementation of fixed-length template protection based on homomorphic encryption with application to signature biometrics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 191–198.

[104] T. Connie, A. Teoh, M. Goh, and D. Ngo, "Palmhashing: A novel approach for cancelable biometrics," *Information processing letters*, vol. 93, no. 1, pp. 1–5, 2005.

[105] A. Teoh, Y. W. Kuan, and S. Lee, "Cancellable biometrics and annotations on biohash," *Pattern recognition*, vol. 41, no. 6, pp. 2034–2044, 2008.

[106] O. Ouda, N. Tsumura, and T. Nakaguchi, "Bioencoding: A reliable tokenless cancelable biometrics scheme for protecting iriscodes," *IEICE Transactions on Information and Systems*, vol. 93, no. 7, pp. 1878–1888, 2010.

[107] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE Symp. on Security and Privacy*, 2017.

[108] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1322–1333.

[109] D. Tang, Z. Zhou, Y. Zhang, and K. Zhang, "Face flashing: A secure liveness detection protocol based on light reflections," *NDSS*, 2018.

[110] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE Symposium on Security and Privacy*, 2016, pp. 582–597.

[111] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv:1706.06083*, 2017.

[112] A. Mittal, A. K. Moorthy, and A. C. Bovik, "No-reference image quality assessment in the spatial domain," *IEEE Transactions on Image Processing*, vol. 21, no. 12, pp. 4695–4708, 2012.

[113] J. Chen, Y. Deng, G. Bai, and G. Su, "Face image quality assessment based on learning to rank," *IEEE signal processing letters*, vol. 22, no. 1, pp. 90–94, 2014.

[114] L. Best-Rowden and A. K. Jain, "Automatic face image quality prediction," *arXiv:1706.09887*, 2017.

[115] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S. Chang, and T. Sainath, "Deep learning for audio signal processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, 2019.

[116] E. T. Affonso, R. D. Nunes, R. L. Rosa, G. F. Pivaro, and D. Z. Rodriguez, "Speech quality assessment in wireless voip communication using deep belief network," *IEEE Access*, vol. 6, pp. 77 022–77 032, 2018.

[117] A. R. Avila, H. Gamper, C. Reddy, R. Cutler, I. Tashev, and J. Gehrke, "Non-intrusive speech quality assessment using neural networks," in *International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2019, pp. 631–635.

[118] J. Ji, J. Li, S. Yan, B. Zhang, and Q. Tian, "Super-bit locality-sensitive hashing," in *Advances in Neural Information Processing Systems*, 2012.

[119] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the ACM symposium on Theory of computing*, 2002, pp. 380–388.

[120] K. Bowyer, K. Hollingsworth, and P. Flynn, "Image understanding for iris biometrics: A survey," *Computer vision and image understanding*, vol. 110, no. 2, 2008.

[121] L. J. Raphael, "Preceding vowel duration as a cue to the perception of the voicing characteristic of word-final consonants in american english," *The Journal of the Acoustical Society of America*, vol. 51, no. 4B, pp. 1296–1303, 1972.

[122] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," *arXiv:1706.08612*, 2017.

[123] W. J. Scheirer and T. E. Boult, "Cracking fuzzy vaults and biometric encryption," in *Biometrics Symposium*, IEEE, 2007, pp. 1–6.

[124] A. Kholmatov and B. Yanikoglu, "Realization of correlation attack against the fuzzy vault scheme," in *Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, International Society for Optics and Photonics, vol. 6819, 2008.

[125] K. Simoens, P. Tuyls, and B. Preneel, "Privacy weaknesses in biometric sketches," in *IEEE Symposium on Security and Privacy*, 2009, pp. 188–203.

[126] R. Canetti and R. R. Dakdouk, "Obfuscating point functions with multibit output," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2008, pp. 489–508.

[127] J. V. Howard and F. Frazier, *Systems and methods for recognition of individuals using multiple biometric searches*, US Patent 7,277,891, 2007.

[128] H. Saevanee, N. Clarke, S. Furnell, and V. Biscione, "Continuous user authentication using multi-modal biometrics," *Computers & Security*, vol. 53, 2015.

[129] *Combining biometrics*, http://www.cl.cam.ac.uk/users/jgd1000/combine/, 2020.

[130] A. Ross and A. K. Jain, "Multimodal biometrics: An overview," in *2004 12th European Signal Processing Conference*, IEEE, 2004, pp. 1221–1224.

[131] L. Hong and A. K. Jain, "Integrating faces and fingerprints for personal identification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 12, pp. 1295–1307, 1998.

[132] Y. A. Zuev and S. Ivanov, "The voting as a way to increase the decision reliability," *Journal of the Franklin Institute*, vol. 336, no. 2, pp. 361–378, 1999.

[133] J. Fierrez-Aguilar, Y. Chen, J. Ortega-Garcia, and A. K. Jain, "Incorporating image quality in multi-algorithm fingerprint verification," in *International Conference on Biometrics*, Springer, 2006, pp. 213–220.

[134] A. Jin, D. Ling, and A. Goh, "Biohashing: Two factor authentication featuring fingerprint data and tokenised random number," *Pattern recognition*, vol. 37, no. 11, 2004.

[135] F. Hao, R. Anderson, and J. Daugman, "Combining crypto with biometrics effectively," *IEEE transactions on computers*, vol. 55, no. 9, pp. 1081–1088, 2006.

[136] N. Sakimura, J. Bradley, M. Jones, B. De Medeiros, and C. Mortimore, "Openid connect core 1.0 incorporating errata set 1," *The OpenID Foundation*, 2014.

[137] Information technology — Security techniques — Biometric information protection, "Iso/iec 24745:2011," *ISO/IEC JTC 1/SC 27 Information security, cybersecurity and privacy protection*, 2011.

[138] M. Gomez-Barrero, J. Galbally, C. Rathgeb, and C. Busch, "General framework to evaluate unlinkability in biometric template protection systems," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 6, 2017.

[139] M. Blanton and M. Aliasgari, "On the (non-) reusability of fuzzy sketches and extractors and security in the computational setting," in *Proceedings of the International Conference on Security and Cryptography*, IEEE, 2011, pp. 68–77.

[140] ——, "Analysis of reusability of secure sketches and fuzzy extractors," *IEEE transactions on information forensics and security*, vol. 8, no. 9, pp. 1433–1445, 2013.

[141] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," in *European Conference on Computer Vision*, Springer, 2016, pp. 87–102.

[142] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 529–534.

[143] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2015, pp. 5206–5210.

[144] Android Open Source Project, *Biometric security*, https://source.android.com/security/biometric/measure, 2020.

[145] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," *Advances in neural information processing systems*, vol. 22, pp. 1509–1517, 2009.

[146] S. Owre, J. M. Rushby, and N. Shankar, "Pvs: A prototype verification system," in *International Conference on Automated Deduction*, Springer, 1992, pp. 748–752.

[147] F. Wolf, R. Kuber, and A. J. Aviv, "" pretty close to a must-have" balancing usability desire and security concern in biometric adoption," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–12.

[148] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[149] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Security and Privacy (SP), 2012 IEEE Symposium on*, IEEE, 2012, pp. 553–567.

[150] L. O'Gorman, "Comparing passwords, tokens, and biometrics for user authentication," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2021–2040, 2003.

[151] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[152] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[153] D. Dowson and B. Landau, "The fréchet distance between multivariate normal distributions," *Journal of multivariate analysis*, vol. 12, no. 3, pp. 450–455, 1982.

[154] The Intercept, https://theintercept.com/2018/05/30/face-recognition-schools-school-shootings/, Dec. 2020.

[155] The Verge, *Moscow's facial recognition system can be hijacked*, https://www.theverge.com/2020/11/11/21561018/moscows-facial-recognition-system-crime-bribe-stalking, Dec. 2020.

[156] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, "Efficient batched oblivious prf with applications to private set intersection," in *CCS*, 2016.

[157] B. Pinkas, T. Schneider, C. Weinert, and U. Wieder, "Efficient circuit-based psi via cuckoo hashing," in *EUROCRYPT*, 2018.

[158] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *CCS*, 2017.

[159] H. Chen, Z. Huang, K. Laine, and P. Rindal, "Labeled psi from fully homomorphic encryption with malicious security," in *CCS*, 2018.

[160] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *EUROCRYPT*, 2004.

[161] L. Chmielewski and J.-H. Hoepman, "Fuzzy private matching," in *ARES*, 2008.

[162] Q. Ye, R. Steinfeld, J. Pieprzyk, and H. Wang, "Efficient fuzzy matching and intersection on private datasets," in *ISISC*, 2009.

[163] I. Calapodescu, S. Estehghari, and J. Clier, *Compact fuzzy private matching using a fully-homomorphic encryption scheme*, US Patent 9,749,128, Aug. 2017.

[164] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *PETS*, 2009.

[165] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *ICISC*, 2009.

[166] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, *et al.*, "Privacy-preserving fingercode authentication," in *MM&Sec*, 2010.

[167] M. Blanton and P. Gasti, "Secure and efficient protocols for iris and fingerprint identification," in *ESORICS*, Springer, 2011.

[168] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich, "Scifi-a system for secure face identification," in *2010 IEEE Symposium on Security and Privacy*, IEEE, 2010, pp. 239–254.

[169] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits.," in *USENIX*, 2011, pp. 331–335.

[170] M. Yasuda, "Secure hamming distance computation for biometrics using ideal-lattice and ring-lwe homomorphic encryption," *Information Security Journal: A Global Perspective*, vol. 26, no. 2, pp. 85–103, 2017.

[171] J. Yuan and S. Yu, "Efficient privacy-preserving biometric identification in cloud computing," in *IEEE INFOCOM*, 2013.

[172] Q. Wang, S. Hu, K. Ren, M. He, M. Du, and Z. Wang, "Cloudbi: Practical privacy-preserving outsourcing of biometric identification in the cloud," in *ESORICS*, 2015.

[173] Y. Zhu, Z. Wang, and J. Wang, "Collusion-resisting secure nearest neighbor query over encrypted data in cloud, revisited," in *IEEE/ACM IWQoS*, 2016.

[174] C. Zhang, L. Zhu, and C. Xu, "Ptbi: An efficient privacy-preserving biometric identification based on perturbed term in the cloud," *IS*, 2017.

[175] L. Zhu, C. Zhang, C. Xu, X. Liu, and C. Huang, "An efficient and privacy-preserving biometric identification scheme in cloud computing," *IEEE Access*, vol. 6, pp. 19 025–19 033, 2018.

[176] M. S. Riazi, B. Chen, A. Shrivastava, D. S. Wallach, and F. Koushanfar, "Sub-linear privacy-preserving search with untrusted server and semi-honest parties," *CoRR*, 2016.

[177] M. Kuzu, S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *IEEE ICDE*, 2012.

[178] A. Boldyreva and N. Chenette, "Efficient fuzzy search on encrypted data," in *International Workshop on FSE*, Springer, 2014.

[179] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 604–613.

[180] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *IEEE ICCV*, 2009, pp. 2130–2137.

[181] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov, "Hamming distance metric learning," in *Advances in neural information processing systems*, 2012, pp. 1061–1069.

[182] X. Yi, C. Caramanis, and E. Price, "Binary embedding: Fundamental limits and fast algorithm," in *ICML*, 2015, pp. 2162–2170.

[183] Y. Vizilter, V. Gorbatsevich, A. Vorotnikov, and N. Kostromov, "Real-time face identification via cnn and boosted hashing forest," in *IEEE CVPR Workshops*, 2016, pp. 78–86.

[184] R. Ji, H. Liu, L. Cao, D. Liu, Y. Wu, and F. Huang, "Toward optimal manifold hashing via discrete locally linear embedding," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5411–5420, 2017.

[185] Z. Dong, C. Jing, M. Pei, and Y. Jia, "Deep cnn based binary hash video representations for face retrieval," *Pattern Recognition*, vol. 81, 2018.

[186] J. Wang, T. Zhang, N. Sebe, H. T. Shen, *et al.*, "A survey on learning to hash," *IEEE TPAMI*, vol. 40, no. 4, pp. 769–790, 2017.

[187] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[188] R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. Smith, "Reusable fuzzy extractors for low-entropy distributions," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2016, pp. 117–146.

[189] X. Wang, A. J. Malozemoff, and J. Katz, *EMP-toolkit*, https://github.com/emp-toolkit, 2016.

[190] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *FnT Privacy and Security*, vol. 2, 2018.

[191] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold, "Keyword search and oblivious pseudorandom functions," in *TCC*, 2005.

[192] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *TOCT*, vol. 6, no. 3, pp. 1–36, 2014.

[193] N. P. Smart and F. Vercauteren, "Fully homomorphic simd operations," *Designs, codes and cryptography*, vol. 71, no. 1, pp. 57–81, 2014.

[194] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private set intersection using permutation-based hashing," in *USENIX*, 2015.

[195] B. Pinkas, T. Schneider, and M. Zohner, "Faster private set intersection based on OT extension," in *USENIX*, 2014, pp. 797–812.

[196] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the aes circuit," in *Annual Cryptology Conference*, Springer, 2012, pp. 850–867.

[197] Z. Brakerski, C. Gentry, and S. Halevi, "Packed ciphertexts in lwe-based homomorphic encryption," in *International Workshop on Public Key Cryptography*, Springer, 2013, pp. 1–13.

[198] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, PMLR, 2016, pp. 201–210.

[199] L. Ducas and D. Stehlé, "Sanitization of fhe ciphertexts," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2016, pp. 294–310.

[200] A. Shamir, "How to share a secret," *Commun. ACM*, 1979.

[201] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. J. Strauss, and R. N. Wright, "Secure multiparty computation of approximations," *ACM Trans. Algorithms*, vol. 2, no. 3, pp. 435–472, Jul. 2006.

[202] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption.," *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012.

[203] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning.," in *AAAI*, vol. 4, 2017, p. 12.

[204] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[205] E. Meijering, "A chronology of interpolation: From ancient astronomy to modern signal and image processing," *Proc. IEEE*, 2002.

[206] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *IEEE WACV*, 1994.

[207] A. Babenko and V. Lempitsky, "Efficient indexing of billion-scale datasets of deep descriptors," in *IEEE CVPR*, 2016.

[208] P. Grother, P. Grother, M. Ngan, and K. Hanaoka, *Face recognition vendor test (frvt) part 2: Identification*. NIST, 2019.

[209] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *Journal of Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015.

[210] Y. Huang, L. Malka, D. Evans, and J. Katz, "Efficient privacy-preserving biometric identification," in *NDSS*, 2011.

[211] E. M. Songhori, S. U. Hussain, A.-R. Sadeghi, and F. Koushanfar, "Compacting privacy-preserving k-nearest neighbor search using logic synthesis," in *IEEE DAC*, 2015.

[212] D. Demmler, T. Schneider, and M. Zohner, "Aby-a framework for efficient mixed-protocol secure two-party computation.," in *NDSS*, 2015.

VITA

Erkam Uzun was born in Istanbul, Turkey in 1988. He is married with Gozde, and they have a daughter, Merih Sara. Erkam joined the Institute for Information Security and Privacy at Georgia Tech in 2015, and conducted research on security and privacy of biometrics systems under the supervision of Prof. Wenke Lee. Before joining Georgia Tech, he worked in Center for Cyber Security at New York University Abu Dhabi as a Research Engineer for two years. He currently holds two B.Sc. degrees in both Computer Engineering and Electrical & Electronics Engineering, and an M.Sc. degree in Computer Engineering from TOBB University of Economics and Technology in Ankara, the capital city of Turkey. His research interests span a broad range of topics including applied cryptography, security/privacy and forensics. His research has resulted in numerous awards, patents, licences, and publications that have appeared in top-tier venues and various news outlines.